AR-010-153

DSTO-GD-0129

A Guide to Using Suppressor in Studies of Large Scale Air Operations

Robert Whitehurst, Jane Phipps and Victor Kowalenko

DEPARTMENT ◆ OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

A Guide to Using Suppressor in Studies of Large Scale Air Operations

Robert Whitehurst, Jane Phipps and Victor Kowalenko

Air Operations Division
Aeronautical and Maritime Research Laboratory

DSTO-GD-0129

ABSTRACT

This guide is an introduction and reference to the Suppressor Simulation System, a powerful and flexible computer code which allows models of integrated military operations to be put into an Australian context. Such mission level models are vital ingredients to models of entire military campaigns. This guide describes in detail how such models are constructed using Suppressor and also provides a comprehensive reference to the Suppressor command language to assist its use in AOD.

RELEASE LIMITATION

Approved for public release

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Victoria 3001

Telephone: (03) 9626 7000
Fax: (03) 9626 7999
© Commonwealth of Australia 1997
AR-010-153
February 1997

A Guide to Using Suppressor in Studies of Large Scale Air Operations

Executive Summary

This guide serves as both a 'user friendly' introduction to the Suppressor Simulation System, an extensive and flexible computer code for the modelling large scale military operations. This simulation system, called Suppressor by its U.S. authors, was initially commissioned to study Warsaw Pact penetration of NATO air defences. Suppressor's importance to AMRL lies in its ability to model the interactions of systems whose properties are defined by the analyst. Because of this military scenarios modelled with Suppressor can employ ADF equipment in an Australian operational context. This guide is the first stage of a programme of capability development which will allow the AOD to use Suppressor as a mission level modelling tool for sponsored Tasks.

This guide is to be used in conjunction with the companion 'Programmer's Reference Manual' which gives a comprehensive hierarchical description of the Suppressor language.

The design of Suppressor means that is particularly well suited for examining mission level models. Using Suppressor will allow studies to concentrate not on the specific performance of particular systems used in isolation, but rather to study how well they operate in integrated environments. Using Suppressor allows models of the combined effectiveness of new military hardware or changed C^{3}I structures. Furthermore these models can include other existing equipment and command structures and place the whole in an operational context. With the help of insights gained from such studies a clearer picture can be drawn of the advantages, or disadvantages, of using or acquiring complex military equipment.

Suppressor is a valuable tool for modelling medium and large scale military operations in an Australian context. It allows the modelling of many different complex scenarios using an essentially constant set of characteristics which define the component systems taking part in these scenarios. Because a high order of physical fidelity can be captured by the model highly discriminatory measures of effectiveness (MOEs) can be deduced from these scenarios. These MOEs can in turn be used to analyse the effectiveness of component systems themselves, or the way in which they are used and controlled, or some combination of the above. Finally the results of Suppressor's mission level models can be combined to provide detailed models of entire military campaigns lasting from days to weeks when they are synthesized with other, campaign level, models. In this way Suppressor can be used to play a part in an overall model of 'Short Warning Conflicts', an area of considerable importance to Australian defence.

Contents

1. Introduction

The purpose of this guide is to introduce the new user to the capabilities of Suppressor, a very sophisticated FORTRAN code that solves problems connected with the modelling of military missions. This program, which consists of several hundred program units, allows users to define the characteristics, interactions and inter-relationships of participants or players in a multi-sided conflict involving air, ground, naval and space forces to many levels of detail. It is particularly useful in mission level studies of weapon systems, platforms, Electronic Warfare (EW) suites and Command, Control, Communications and Intelligence (C^3I) architectures. It has the capability to simulate missions with durations lasting from a few minutes up to several days. Suppressor treats the interactions of forces in less detail than most one-on-one engagement models, it comes into its own when simulating raid, mission and campaign level scenarios. Although Suppressor is not suitable for multi-day conflicts where tactics change continuously it is capable of modelling scenarios spanning almost half the globe. Suppressor is not an interactive program, rather its simulations are based on suitably prepared user input data.

Before Suppressor can be employed as an analytical tool for Australian Defence Force (ADF) scenarios, the user must determine the scope of the problem, which may range in complexity from a simple back of the envelope calculation to one requiring large computational resources for fine-grained modelling. Next, the systems involved in the problem must be examined in terms of their roles in a hierarchy of models beginning with one-on-one situations, then progressing to mission level and finally to campaign level situations. For example, to model an aircraft system one would commence with one-on-one models of engagements, whose output would be conditional probability of kill for each engagement, P_K. These would address whether a kill would occur given the specific conditions under which a weapon was fired. These probabilities would then serve as input to a raid or single mission level model such as Suppressor. Suppressor itself is not suitable for determining these P_K tables, but rather is suitable in determining how effective a weapon is in a realistic military scenario. This is because Suppressor can handle an entire mission consisting of a number of systems as they move through a simulated environment. Its output would include results from various mixes of forces perhaps involving hundreds of systems. This output, namely the outcome of various military missions, can itself be used as input to a suitable campaign model such as Thunder. The campaign level model itself may be used to analyse force effectiveness. As such Suppressor can be seen to fill a central niche in the hierarchy of models, one which allows a bridge between low-level models examining the performance of individual systems in great detail and high level models, which paraphrase system performance in favour of modelling the large scale results of military campaigns.

Given a specific problem the first question that should be asked is whether this problem is indeed suitable for modelling with Suppressor. If the problem is one of determining the physical capabilities of some system such as an aircraft or a weapon then the answer will clearly be no. If, on the other hand, the problem is to model a large scale conflict running over several weeks or months then, although Suppressor is not appropriate for the overall problem, it may well be useful in determining the outcomes of individual missions and raids that occurred within this conflict. If the problem is examining the effectiveness of some military system when used in realistic combat roles, then Suppressor is likely to be an appropriate means of carrying out the study.

Once it seems reasonable that Suppressor is suitable for the problem an appropriate scenario must still be designed. Modelling a specific combat problem may be divided into six steps.

- Firstly the major issues, which may perhaps be attrition of aircraft, should be determined. These criteria alone will not normally be sufficient to design a relevant model.
- The number of viable models can be reduced in the second step by defining sub-issues to a sufficiently detailed level where evaluation criteria can be identified. For example, sub-issues for the attrition of aircraft could be: what resources were expended in causing the attrition, how many targets were hit before an aircraft was destroyed.
- In the next step each sub-issue is assigned a set of numbers known as Measures of Effectiveness (MOEs) that correspond to each particular sub-issue. For example, if the user wants to know how many aircraft were destroyed, then the MOE for this sub-issue is simply the number of aircraft destroyed, which may be before or after the weapon was released. At this stage it is appropriate to check whether or not Suppressor can provide the specific information required to compute the MOE.
- For the fourth step the user selects the appropriate models that both address the major issues and produce the MOEs for the specific sub-issues. In addition, other factors such as whether the chosen model can be run on the available machine need to be considered at this point.
- In the fifth step the user examines what input data are required for the selected models and then must determine whether the data can be collected. This is a critical step because the level of detail the user provides as input has a major effect on the output and hence, the overall validity of the model.
- The final step in this planning process requires that the user identify data sources for the input to the various systems.

Suppose, for example, a problem posed is to consider the effectiveness of an EF-111A Raven in defence suppression. Since the EF-111A is a jamming aircraft, it contributes to non-lethal defence suppression. That is, its role is to stop the transfer of information enabling a defensive system to launch a weapon at a target. This main issue can be broken down into sub-issues. Examples of sub-issues include measures of by how much the defensive force's ability to transfer information is denied, delayed or degraded. Thus the three processes of denying, delaying and degrading each represent sub-issues within the broader issues of nonlethal defence suppression that are relevant to the main question of the Raven's effectiveness.

As mentioned above sub-issues are evaluated by assigning sets of MOEs to them. For the example above, one might wish to determine how well the Raven affects the enemy's ability to detect, lock-on, engage and kill targets. These questions can be answered by comparing the MOE set that enumerates these abilities both before and after jamming was employed. Possible MOEs may include the following:

- reductions and delays in enemy activity, in particular the number of detections and lock-ons the enemy is able to achieve in a given time,
- changes in defence capability, such as how many systems were required to accomplish a mission or what losses were experienced,
- changes in attack effectiveness, such as the ratios of air-to-air exchanges or how many aircraft one side lost.

Selection of a model that is sensitive to the chosen MOEs will depend critically on the level of detail at which output is to be generated, which in turn depends on the level of input detail. A system's level of detail can be either notional or explicit, the former referring to a situation in which an entity or process is vague or abstract and the latter to the situation where an entity or process is described in great detail. These two cases are examples of low and high fidelity modelling. Typically models of scenarios will include both notional and explicit representations of entities and processes. Notional and explicit levels of detail apply to both specific physical components of the scenario and more abstract features such as their interactions and behaviour in certain circumstances. Furthermore the influence of the human component of the scenario may also be modelled with various degrees of fidelity. It is also important that results obtained from the model do not depend too strongly on the level of detail employed. If the results are effected the obtained MOEs should be regarded as being quite unreliable unless their are good reasons for the contrary. It follows that a good study will establish that the MOEs obtained are reproducible in models of varying fidelity.

A natural way of obtaining results from models with varying proportions of notional and explicit detail is that initially models with fairly low fidelity be run and analysed. As these results are obtained the models may be refined until the final stage of the development is reached, with intermediate results being obtained as the process of model development progresses.

There are many specific methods of varying the model's fidelity within Suppressor. For example, systems can be modelled either individually, such as the actions of a single aircraft throughout a mission; or systems may be grouped at a coarser level of detail. Here all aircraft within a strike may be considered as a single unit. Physical characteristics can also be notional, for example, modelling the shape of an aircraft as a rectangle, or specific in which case the aircraft's shape would be modelled as accurately as possible using its true physical dimensions. In the former case the cross-section of the aircraft as seen by both radars and other sensors would only approximate that of a real aircraft, while in the latter its radar and optical signature would be rather accurate. It is interesting to note here that it is pointless to accurately model the aircraft's cross-section if the sensors are only crudely represented. It follows that it not always sensible to vary the fidelity of a single component in a simulation and expect radically improved accuracy to result.

Weapon systems provide a another good example of varying the detail of the model. At the coarsest level a missile may be modelled as a simple weapon which when fired at a given range and orientation from the target has a certain probability of kill. On the other hand the missile may be modelled as a complete player within the scenario, with its own physical and operational characteristics, guidance systems tactics and warhead.

A simple military model may be carried out on a smooth surface devoid of natural features such as lakes, mountains, seas and valleys. However, Suppressor can process and integrate digital terrain data in the form of Defense Mapping Agency level one DTED, (Digital Terrain Elevation Data). This allows explicit topographic detail to also be added to a scenario. This will of course greatly improve the accuracy of models involving low flying strike aircraft attempting to evade detection on their approach.

It should be mentioned that in addition to mobile systems such as aircraft, satellites, ships, tanks and so on fixed systems such as SAM sites, command posts, supply depots and bridges can also be modelled in Suppressor. Thus attrition analyses for both targets and attackers can be obtained from the code. In particular, the code is sensitive to a wide range of factors affecting the outcome of a scenario including speeds, altitudes, electromagnetic signatures, sensors, weapons, tactics, communications, electronic countermeasures (ECM), rules of engagement, the physical environment and available resources. In Suppressor tactics, rules of engagement, and command and control are part of the input unlike other models such as EADSIM where these are hard-coded into software.

At this point the discussion is leading us to consider the actual nature of Suppressor rather than merely introducing the user to its capabilities. We have already implied that Suppressor models military systems such as aircraft or SAM sites. However, as noted above, there are no pre-packaged systems provided within Suppressor. There are no ready to use components labelled 'fighter aircraft' or 'destroyer'. Nor indeed are the systems that may be found within these entities provided with Suppressor, there are no 'radars' or 'guided missiles' included as part of the Suppressor package. Instead Suppressor provides a kind of kit of building blocks out of which these systems and pieces of equipment can be assembled by the user. Suppressor may be viewed as a box of lego bricks out of which elaborate models of real systems can be constructed.

The bricks which Suppressor provides consist of six 'generic functions' which completely describe all the essential capabilities of most military equipment. These consist of five physical functions, which define what a particular system can do, and a sixth abstract function of 'thinking' which describes how the system processes its perceptions of its environment and how it makes decisions. The generic functions are:

- Moving, which is required by all systems that move.
- Talking, which is needed whenever a system is to communicate with another.
- Sensing, which allows a system to detect other systems, for example with a radar.
- Shooting, which allows a system to use weapons.
- Disrupting, which enables a system to jam or interfere with other systems.
- Thinking, which models the human (or machine) decision making process. In practice almost all modelled entities will require the ability to think, since even the simplest system needs to perceive its environment and in Suppressor all interactions with the environment are explicitly modelled as part of a mental process.

By combining the capabilities endowed by these 'bricks' the user can model sophisticated military equipment to match her or his own needs. This is a particular attractive feature of Suppressor, since it allows scenarios to be constructed which are appropriate to Australia.

To clarify the above, consider the generic functions characterising an airborne early warning (AEW) craft. In addition to moving and sensing, such an aircraft might be required to talk in order to send and receive assignments. It would clearly need to be able to 'think' to respond to a changing situation and to follow its own orders and plans. The resulting model of an AEW craft has the capability to perform four of the six generic functions. Such a system, built up out of components is called a 'player-type' by Suppressor. Other player-types might include a static command post having no other capabilities than to think and talk. The previously mentioned example of a Raven EF-111A would have the ability to disrupt targets as well as the ability to move, sense, talk and think. A fighter aircraft would perform the same functions as the AEW craft in addition to shooting, an ability that an EF-111A would not possess.

Player-types form templates from which many copies representing individual players may be struck within a single scenario. So if a scenario called for six strike aircraft of identical capabilities it is not necessary to define each one separately, instead a single player-type is designed and six copies struck to represent the individual players. Although players are modelled according to their type in Suppressor, it is not always possible to distinguish different player-types by their generic functions alone. That is to say in Suppressor different player-types can possess the same generic functions. For example, a naval destroyer could possess the same generic functions of moving, thinking, sensing and shooting as a fighter aircraft. Furthermore a reconnaissance satellite might have the same functions as an AEW craft. These player-types would be distinguished by their input data, which would consist of different values and units, and by their tactics which would be quite different.

The building bricks which Suppressor provides for the construction of a player-type come in two principal varieties. The first of these are component systems which possess the ability to perform specific generic functions. For example a 'sensor' system is able to sense, a 'mover' system to move and so on. The second principal type of building brick are the rules which describe when and how these systems will be deployed by the player-type. These rules are the tactics that control each player-type; the tactics must be processed by a 'thinker' system belonging to the player-type. Typically the thinker, or thinkers, associated with each player-type represent human operators or crew. The generic function of thinking and the tactics associated with it provide the essential link between the player-type and the scenario within which it operates.

The ability to construct the components of a scenario according to the analyst's specifications gives enormous flexibility in the development of Suppressor models. The individual systems used by each player figuring in a scenario can be specified and modelled to closely match the systems to be used in real life. For the AEW craft mentioned above the necessary sub-systems are a radar sensor for sensing; a communication receiver and transmitter for talking; an airframe or body for moving and a thinker for orchestrating the whole ensemble. Similarly, the EF-111A would require an ECM jammer for disrupting, whilst the fighter aircraft needs a weapon system for shooting.

When modelling a scenario using Suppressor three distinct stages occur. The first stage is a data preparation phase wherein all the components of a scenario are synthesized. This stage is where the analyst's model is converted, via a prescribed instruction set, into an internal format Suppressor can understand and process. The second stage is a model execution stage, where the model is run using Suppressor's internal logic. This results in a large record of all the significant incidents that occurred in the simulation. This chronological record may be examined in the third and final stage using post-processing analysis tools in order to extract the MOEs relevant to the study. Yet another, fourth stage, can now be seen wherein the MOEs are interpreted and their implications discussed. This stage, though, is not strictly a part of the process which involves Suppressor.

Distinct instruction sets control each stage of the above process which uses Suppressor. The most complex of these data-sets are those concerned with the preparation of the scenario. It is in this stage that the user's model is translated into a form that Suppressor can use. The fact that Suppressor is so flexible means that an enormous amount of detail is required to construct even a quite modest military model. There are actually five different instruction sets that control the collation of data. The first of these is the 'User Application Names' (UAN) database. The UAN contains the names of all the components that are to going to be defined by the user for inclusion in the model. These components will include such things as the names of the sides in the conflict, the player-types involved along with the names of their constituent elements and systems. The component systems will include items such as radar systems and weapons, the names given to tactics, identification labels for command and engagement zones, the even names of the command chains which are needed by the model. In addition more information, such as the labels of checkpoints and radar cross-sections used by player-types, must be included.

The second instruction set is named the 'Type Data Base' (TDB). In this data-set the characteristics and capabilities of all the component player-types involved in the scenario are to be found. This necessitates that all these player-type's subsidiary systems be defined here too, along with the tactical rules which describe the player-type's operation and the use of their systems. Ideally the TDB would contain no material that is specific to any one scenario. In this way a player-type or a system defined within the TDB can easily be re-used in another scenario. In practice some small changes to a player-type's tactics may be required when used in a different context. Despite this the TDB can be regarded as a library of player-types which may be used and re-used in many different military models. The TDB is the most complicated and time-consuming of all the instruction sets to construct but once data have been collected for a particular player type they can be employed in other scenarios. For example a particular scenario might need all the performance characteristics and capabilities of the AIM 9 missile, which could then be used in subsequent scenarios in quite different contexts. The initial investment in constructing valid and comprehensive TDBs can be very large, but once successfully completed they will be available for many different studies. The TDB, due to the great detail needed to model player-types and their components, will typically be the largest data-set required for any given Suppressor model. A TDB that is still under development may be useable at a lower fidelity for initial analysis of a scenario using notional detail.

The third and fourth data-sets are both used to define the topography of the conflict zone incorporated within the model. As noted above Suppressor can operate with no topographic information at all, in which case all operations take place above what is effectively a smooth seascape. Hence these two data-sets: the 'Defense Mapping Agency' (DMA) data-set and the 'Environmental Data Base' (EDB) may be omitted. When topographic information is desired Suppressor may incorporate level 1 DTED with the use of the DMA data-set. This is essentially a pre-processing step which constructs a terrain data-file in a form useful to Suppressor, and which may then be consolidated using the EDB's instructions. The DMA step is useful due to the wide availability of data in the DTED format, and thus saves the analyst the difficulty of preparing terrain data in the form that may be read in by the EDB. It should be noted that the incorporation of topographic data usually requires a large amount of memory when running a simulation.

The next instruction set, known as the 'Scenario Data Base' (SDB) is the final preparatory data-set. This contains, as the name suggests, information that is specific to each scenario. So information outlining the order of battle and deployment of each side in the conflict, along with the (envisioned) movement plans of each player are to be found in this data-set. Note that here player-types will not occur, but rather the player's that are struck from the player-type templates found in the TDB. A template may be used to create many players, perhaps found on all sides in the conflict. Care should be used in constructing the player-type's tactics to ensure that the behaviour of the players is appropriate for the side to which they belong. Furthermore the SDB contains the location of player's checkpoints and geographical zones, as well as the number and frequencies of communication nets and other scenario dependent items. A single study will normally use several scenarios to model a problem, with each scenario corresponding to a single replication within the study's test matrix. It would therefore be likely that well designed studies will require several different SDB files, one for each scenario under examination.

The second and third stages in the execution of a Suppressor model are, from the user's point of view, much simpler than the first. Each stage has only one instruction set associated with it, and in both cases these data-sets are short and relatively straightforward. The execution phase is controlled by the 'Model Kickoff' (MKO or MOD) data-set. Most of the instructions within this data-set are used to control the precise form of the chronological record of the scenario which is printed out as a record.

The final stage is the analysis or post-processing of the record output by the execution stage. This is controlled with the 'Analysis Data Base' (ADB), which specifies the data that are to be summarized from the (possibly enormous) complete listing that the MOD provided. This final stage is of course optional, and different analysis techniques used upon the data output from running the simulation.

The remainder of this guide is arranged as follows. In section 2 we give an overview of Suppressor's structure and design philosophy, along with a description of how information is passed between the various stages of a Suppressor run. Section 3 is dedicated to the UAN and TDB, and is the most complex and lengthy of the sections. Section 4 explains how to incorporate terrain information into a scenario using the DMA and EDB. The scenario specific SDB is the second most complicated instruction set to construct after the TDB and is presented in section 5. We describe how the MOD may be used to control the execution of the model in section 6. Section 7 is used to describe the ADB and to explain how post-processing may be employed to extract extra information from Suppressor. The guide concludes in section 8 by summarising the major features of Suppressor. Accompanying this guide is a reference manual, the 'Reference Manual to the Suppressor Simulation System'[1]. This manual describes the keywords and their options in detail. Unlike this guide, which is meant to be a introductory document to Suppresor the manual serves as a programmer's ready reference. It will be referred to from time to time in this guide as 'the reference manual'.

[1] A Reference Manual to the Suppressor Simulation System, R. Whitehurst, J. Phipps and V. Kowlanko. DSTO General Document, (1997). [DSTO-GD-0130]

2. Overview of Suppressor

This section seeks to describe the general principles involved in the generation of results using Suppressor. This breaks down into two parts: the transformation of user instructions into Suppressor's internal data format, (the data preparation stage), and the process by which Suppressor advances a simulation. Subsequent sections will discuss the actual data-sets in considerable detail, so here the discussion will be limited to the overall design of Suppressor.

The arena within which a scenario will occur spans both time and space. Suppressor models space as a three dimensional volume, with for most purposes the Earth's surface assumed to be flat. Suppressor's design assumes that in most circumstances distances along the surface of the Earth considerably exceed differences in height. Because of this computations of distances are made in the first instance as ground separations; only later is the vertical component introduced as a correction in cases which require it. This strategy implies that some distortions and inaccuracies result when scenarios span a substantial portion of the globe. Indeed if more than a single hemisphere is included in the model the projected ground positions of the two hemispheres will overlap. It should be noted that line-of-sight computations do allow a horizon to be included as a simple circle of the appropriate radius around the current view-point of a player. In addition Suppressor may superimpose terrain information on this 'flat Earth' model which allows it to measure altitudes both 'above ground level' and above 'mean sea level'. Suppressor has a fundamental spatial resolution of one metre, even though positions may be represented as floating point numbers to a higher accuracy than this.

Events occurring within a scenario occur at a modelled 'game time'. This is the time elapsed since some reference time which is usually the beginning of the scenario although the scenario may commence some time after this moment. This may be convenient when comparing the effectiveness of two scenarios differing in part by their time of commencement. For example, some mission may start at 'D plus fifty minutes' where D is the reference time. The maximum temporal resolution of Suppressor is limited only by the resolution of the single precision real numbers used to encode them. This usually implies that the shortest time resolvable in a model will be about 10^{-7} of the total length of a mission. So if an accuracy of 0.1s is required the whole model should span less than 10 days. If time must be resolved down to units of milliseconds then the whole scenario should not last longer than a couple of hours.

Once the overall scope in space and time of the scenario has been established the next stage is the preparation and processing of the data files which describe the model. These files, the UAN, TDB, DMA, EDB and SDB data-sets describe the scenario completely and are processed, as discussed in the introduction, in a specific order. The data-sets themselves are files with an English language based syntax which specify the user's requirements for the components of the model. Each file is read in turn creating as its output a listing file, again with an English like syntax which can be easily read by the user, and at least one binary data file which, along with subsequent input instruction sets, constitute the input of the next stage of the Suppressor run. It is simple to automatically process the Suppressor input files with a simple program written using a Unix shell script or the appropriate command language.

Although, to the user, the model execution (MKO or MOD) step is very simple, since relatively few user instructions are available to modify it, it is by far the most complex part of the Suppressor model. It is here that the scenario is simulated and its outcomes evaluated.

The execution phase of Suppressor is deterministic, despite the inclusion of some randomized behaviour via pseudo-random numbers. It would therefore be possible, in principle, to anticipate the outcome of any scenario by analysing carefully all the starting conditions and the rules that will be followed during the model run. In practice, however, there are so many interacting components which influence the results obtained that this analysis would be infeasible for all bar the most trivial scenarios. Suppressor arrives at its results by maintaining (and continuously updating) a list of all the 'events' which have are to occur in the model or have already happened. An event is a term that is used to describe an occurrence in the model that can occur instantaneously. Actual 'events' or happenings in the scenario will in practice take some time, but they may be divided into occurrences which are so quick as to be effectively instantaneous, and those with a definite duration. The former are known as incidents and can be described by a single Suppressor event. An example may be the moment of impact of a missile with its target, or the instant in which a radar senses a target. On the other hand anything which takes longer than an instant to occur is known as an 'activity' and is modelled with two Suppressor events, one which commences the activity, the other which ends it. An example of an activity might be the process of thinking, or the action of preparing a weapon to fire.

The above list of events, (the Scenario Action Item List or SAIL), is strictly time ordered and once one event has been processed the next on the list is taken for analysis. The game time advances as each event is reached with, as noted above, the events themselves assumed to be instantaneous. Because of this an event, once selected, becomes decoupled from the rest of the model and can no longer be influenced by anything else in the scenario. This is not the case before the event is processed. This is because the results of a single event may well be to schedule new events for later times and to cancel already scheduled events. If an event, for example, is the destruction of a target by a missile, then future events scheduled for the player representing the target will need to be deleted from the SAIL. Because unforeseen circumstances can change the SAIL, Suppressor schedules as few events as it can. In this way as little modification as possible to the SAIL must be made when players' plans change or players are destroyed. In addition to changing the SAIL an event may cause the model's stored data to be updated. For example, some data in Suppressor records players' perceptions of the current situation. Initially a player may be informed that a target is at some given location. During the scenario it may receive intelligence with a more current position, at which point its 'memory' of the player's position will be altered.

2.1 Event Management in Suppressor

We can now describe, albeit in rather simplistic terms, just how the Suppressor model functions. At the beginning of the MOD stage all the user input data has been processed in the preparatory stages, and Suppressor has used these data to construct the initial SAIL describing the model. This will contain all the initially scheduled events. In addition various data structures will have been created describing the state of the system. For example each player will have knowledge, i.e. perceptions, of the current situation. This information may, or may not, be accurate and it is on the basis of this data that a player bases its actions.

When Suppressor begins it selects the first element of the SAIL, this event will either be a physical event, i.e. events concerned with communication, movement, sensing and firing of weapons; or a mental event which are events that model the thinking processes of a player. There are no physical events explicitly related to disrupting since jamming merely complicates the communications and sensing events. The two classes of events, mental and physical, are handled by separate algorithms but in each case new events may be added to the SAIL, or previously scheduled events cancelled as a result of processing of the current event. Simultaneously the data held within Suppressor may be changed. This process will continue until all events in the SAIL have been processed or the game end time is reached, whichever is the sooner.

The most important processes within Suppressor are the thinking events, these activities are central to the behaviour of the players in the model. As such it is useful to examine in slightly more detail Suppressor's processing of mental events. First of all thinking processes are activities, which take a certain amount of time to occur. These thinking times are in fact defined by the user in the TDB. Each thinker will have associated with it several thinking times which it must possess to be able to process events of the relevant type. For example, the thinking time '`CONSIDER-MOVE`' must be defined to allow a thinker to consider manoeuvring the player away from its pre-programmed path.

The processes that occur when a mental event is processed by Suppressor depend on whether this event starts or stops a thinking activity. Two (instantaneous) events bound all activities, and the event at the start of the thinking activity is known as the 'stimulus' or 'non-decision' event, whilst the event at the end of the activity is named the 'decision' event.

Firstly, consider the consequences of a 'stimulus' event being removed from the SAIL for processing. The event is passed first to the appropriate player's pending queue. Each player has one such queue, and it is used to hold mental activities which awaiting consideration by that player. The next step is to check to see if there is a thinker available to begin thinking about the events on the pending queue. This will be true if there is at least one thinker not currently active which has the appropriate responsibilities. Suppose an event is to do with considering a manoeuvre, then the thinker would need to have the 'CONSIDER-MOVE' responsibility defined by the user in order to process this stimulus. If any matches are made between pending events and thinkers then for each match a new decision event will be scheduled on the SAIL. This is merely the event which will terminate the current thinking activity and will be scheduled after a delay given by the thinking time. The other consequence of the match is that the chosen thinker will no longer be free. Once no more thinkers are available the mental processing will have ended and the next item on the SAIL can be selected by Suppressor.

The second possible mental event is that a decision event is passed from the SAIL to the player. This event is associated with the end of a thinking activity and so will be (already) associated with a specific thinker. This thinker will be relabelled as being free and the consequences of the 'decision' will be examined by Suppressor's logic. As a result new events may be scheduled for the SAIL, or directly for the current player's pending queue, or events may be cancelled from the SAIL. Furthermore, because a thinker is now free, it is necessary to check to see if any stimulus events on the pending queue can be matched with it.

From the above discussion we can see that if all items on the pending queue can always be handled by the player as they arrive, so that no entries remain on this queue, then the player will be able to process all the information it receives. If, however, stimuli start to arrive too rapidly the player's thinkers will become overloaded and unable to cope. In these situations important decisions will be left unmade and as a result mistakes will be likely made.

In Suppressor all the actions undertaken by a player are mediated by that player's mental processes. As an example of how this works in practice consider the events associated with sensors and sensing. To facilitate the computing of sensing probabilities Suppressor maintains a sensor chance list, (SCL), which is simply the list of all players that a particular sensor could possibly detect at any given moment. The relative movement of the sensor and the potential target means that this list must be continuously updated. The SCL is then examined each time a sensing chance is computed for each sensor. A sensing chance is a physical event, and they occur at a user specified rate for each sensor. Each target in the SCL will be checked to see if, at that moment, it can be seen by the sensor. This will depend on factors such as the range of the target, the visibility of the target, the presence of jamming or of intervening terrain and so on. If the chance produces a detection this is recorded by Suppressor in the 'iconic memory' of the player to whom the sensor belongs. To see what happens next then some discussion of the various kinds of memory that each player possesses is necessary.

All information 'known' by a player is recorded in one of several kinds of memory held by that player. All thinkers belonging to that player have access to this knowledge, so that communication between the thinkers of a single player is perfect. The most evanescent of these memories is 'iconic' memory. In the context of a sensor it may be regarded as the phosphor blip on the radar screen. If no notice is taken of this by the operator of the radar, (represented by a thinker), before the next sweep of the radar this information will be lost. So, should the thinker be busy and not paying attention to the new detection and the radar sweeps around the previous sensing chance's result will be lost. The recording of data in iconic memory therefore implies that the information is immediately available for incorporation in the player's environment.

The first thing that may occur after a blip appears on a radar screen is that the operator becomes aware of it. In this case he will have noticed the information. Noticing is the first of the three kinds of thinking process that Suppressor models. In the context of the sensing example the (physical) sensing event will schedule on the SAIL a noticing stimulus event. If this event is associated with the thinker tasked to the sensor before the data disappears from iconic memory, (i.e. before the blip disappears from the screen), then the detection will be noted and the new perception stored in the player's short term memory. This will occur at the end of the noticing activity, which in addition to this operation will schedule a mental 'digestion' event.

Digestion is the second form of thinking modelled by Suppressor. Digestion is the process of incorporating new perceptions into the body of existing perceptions held by a player. These established perceptions form the player's current view of the dynamic aspects of its environment. In other words these data are where and who other players are and what they are currently doing. These perceptions are stored not in short term memory, like the newly noticed perceptions but in medium term memory. The major difference between these memories are that short term memory is more subject to change; new perceptions will be discarded until after some user defined time delay[2]. Medium term memory will retain information of a target beyond this time if the player is currently assigned to act against this target.

Finally once an item has been digested and its significance appreciated various reactions might be scheduled. These are again mental activities, the final form of thinking modelled by Suppressor. Which precise activity depends on the nature of the new perception, but an example might be to the consider manoeuvring to engage the target. In this case a specific stimulus event will be placed on the pending queue of the player, and, should a thinker become free to deal with it, this possibility will be evaluated. The decision taken will depend on the user defined tactics provided in the TDB block, and if the decision to react is made, a physical event will be scheduled. For example the decision to change course or to select a weapon for engagement. These subsequent physical events will appear on the SAIL, with previously planned SAIL events for the player being cancelled if they are no longer relevant.

[2] The TDB item TIME-TO-DROP.

The sequences described above are typical of the sequence of events that occur following a single occurrence, such as a sensing chance. So a physical event will trigger a sequence of responses from a thinker; initially the event itself is stored in the temporary iconic memory. Once the event is *noticed* by a thinker the event is more securely recorded in the players short-term memory. After this the player digests the information, transferring the perception to its medium term memory. At this stage the player will consider *reacting* to the event, maybe triggering one or more subsequent physical events as a consequence. It shows how intimately related physical and mental events are, and how the behaviour of the model depends on Suppressor's model of mental processing. One more class of player memory exists, which has not been mentioned as yet. This is long-term memory, which is essentially permanent. This records the player's knowledge about itself, its own tactics, its own orders and knowledge of other friendly players found along its own chains of command.

2.2 Suppressor's Record of a Scenario

From the perspective of the user the record provided by Suppressor of the events unfolding in the scenario is of vital importance. The information stored in this record will be the primary source for determining the outcome of the simulation and in evaluating the MOEs required for a study.

Whilst it is clearly useless for Suppressor not to record any detail recording too much detail might be almost equally difficult. For example, imagine the enormous difficulties involved if *all* of the above events were to be logged to a data file. Trying to establish why and where who did what to whom would become fiendishly complex. On the other hand it is probably wise for the record to err on the side of verbosity, it being hard to anticipate what every user requires in advance. It would be a shame if the only reason that Suppressor were unsuitable for a study was due to its recalcitrance in revealing what had actually occurred during a scenario.

The solution taken is that Suppressor records most events from the SAIL that it actually carried out in the so-called 'Time History List'. Events that are not recorded are the mental stimuli that commence thinking activities, rather the end of the activity is recorded. Events that only ever figured on the pending queue of a player are not noted.

Rather than simply recording the event Suppressor writes out a short message which indicates also the event's context. For example the physical sensing chance event may be annotated with the entries such as `FIRST-HAS-SENSOR-IN-RANGE-OF`, `CHANGES-IN-DETECTION-FOR` or `RANDOMLY-LOSES-LOCK-ON`, depending on the circumstances. The items are meant to be relatively easily read by a user and usually include extra information giving more details, so that the full entry might look like:

```
5:45.6
10 bomber FIRST-HAS-SENSOR-IN-RANGE-OF 201 target
using 120 bomber_radar_rx sensor (x,y,z): -19.0 13.8 1.200
   km; Spd: 235.0 m/s; Hdg: 82.8 deg
target (x,y,z): 19.500 9.500 0.000 km
2-D distance: 38.7 km; bearing (snr-tgt): 13.64 deg;
```

Which gives the game time, the player name '10 bomber' to whom the sensor belongs, the target which has moved within detection range and so being added to the sensing chance list, '201 target', along with the same of the sensor itself, '120 bomber_radar_rx', and a wealth of information on the positions and relative headings of the two players.

The output records break down into seven functionally related areas: five of these correspond to the physical (or action) generic functions; two to aspects of command and control. The above examples are all related to sensing; other records are related to moving, communications, disrupting, shooting, assignment of subordinates and events related to attacking (lethally engaging) another player.

A complete specification of all the output records is given in the reerence manual. For now it is useful to note that only events that actually occur are recorded. This may seem an obvious point but its consequence is that no output will occur when something doesn't happen due to a backlog of work. In other words it may be hard to tell that a thinker or thinkers have become overloaded and are not keeping up with the volume of work to be done.

The Time History items, which are recorded during the model execution phase may be analysed and summarized with the help of the post-processing ADB phase of the Suppressor run. This optional phase can facilitate the extraction and presentation of results from tested and proven models.

2.3 Suppressor File Dependencies

Having touched upon how Suppressor actually works internally it is now useful to consider the process of preparing a model using Suppressor. As we have noted up to seven different input files must be written, each one with a role to play in translating the selected model into a form understood by Suppressor. Subsequent sections will discuss these instruction sets in detail, but first we shall discuss how these sets interact and communicate with each other.

On the computers of the AOD Suppressor is organized so that all the data files for a single model are to be found under a single directory hierarchy. Suppose the model is named 'aew' then a master directory aew will be created for the model; for example

- Suppressor: Root directory for the Suppressor code.
- Suppressor/bin: directory containing utilities and executable codes.
- Suppressor/lan: The initialization file 'boot.bin' is contained in this directory.
- Suppressor/aew: An example of a model's top level directory, (here named 'aew'). Other directories can be created, one for each required model

Beneath each model directory are several other sub-directories, for example Suppressor/aew/sdb which would contain files connected with the SDB. These directories and their contents are as follows:

- uan: UAN database and files output from the UAN processing stage.
- tdb: TDB instruction set and the TDB listing and binary files.
- dma: DMA instruction set, DMA output files and binary terrain data output by the DMA stage.
- sqr: Files containing 1° square DMA level 1 DTED used to prepare binary terrain files.
- edb: EDB instruction set and files output by the EDB stage, including untranslated binary terrain files.
- sdb: SDB instruction set and the output SDB binary and listing files.
- trn: Binary translated terrain files output by the SDB stage.
- mod: MOD instruction set and the MOD binary and listing files, the latter being the Time History List.
- sit: Binary situation file produced by the MOD stage.
- adb: ADB instruction set and the ADB listing produced by the ADB stage.
- tmp: A work space directory used to hold temporary files created during Suppressor runs.
- grf: Files connected with the 'grafic' graphical post-processor used as a tool for analysing Suppressor runs.

It should be noted that the above directory structure is purely that used at the AOD and may be altered to suit individual requirements. (This being accomplished via the preparation of suitable shell, i.e. batch processing, scripts.)

A naming convention for files is useful, but not essential. At the moment the following suffixes are used to help identify files:

- .dat: User prepared data files, i.e. instruction sets, describing the scenario.
- .lis: Textual listing files produced by Suppressor.
- .bin: Re-locatable binary files produced by Suppressor.
- .dt1: raw 1° square DMA terrain data, (DTED level 1).
- .dma: Terrain data produced by the DMA processing stage of Suppressor.
- .edb: Untranslated binary terrain data produced by the EDB stage.

- .trn: Translated binary terrain data produced by the SDB stage.

Each stage of the Suppressor processing sequence takes one or more input binary files and the user's input instruction file and generates output binary files and an output listing file. The following tables show which input files each processing step requires, and which output files they produce.

LAN

The Suppressor initialization file is created in the 'lan' directory and need not be recomputed for each model; hence it may be kept separately from model specific data and binary files. The output boot file, 'boot.bin' has no file dependencies of its own, so its dependency table is exceedingly simple:

Input Files	Output Files
	lan/boot.bin

UAN

The UAN step may be included as a portion of the TDB step so its presence is not essential. When it is used it will use the files shown below:

Input Files	Output Files
aew/uan/uan_aew.dat	aew/uan/uan_aew.bin
lan/boot.bin	aew/uan/uan_aew.lis

TDB

The TDB stage may either initialize the whole processing sequence of a single model or be a subsequent step to the that of the UAN. In the latter case the files used are:

Input Files	Output Files
aew/tdb/tdb_aew.dat	aew/tdb/tdb_aew.bin
aew/uan/uan_aew.bin	aew/tdb/tdb_aew.lis

If no separate UAN file is used the dependencies change slightly to:

Input Files	Output Files
aew/tdb/tdb_aew.dat	aew/tdb/tdb_aew.bin
lan/boot.bin	aew/tdb/tdb_aew.lis

When more than one user provided TDB instruction set exists and these are to be consolidated into one final TDB binary file during several steps then each intermediate binary file will be passed back into the next TDB step along with the new instruction file. The overall file dependencies listed above will not be changed in this case except that each subsequent TDB step depends on the binary output of the previous step.

DMA

The DMA stage is used to incorporate one degree square DMA level 1 DTED terrain files into a Suppressor scenario. Several contiguous one degree square terrain files found in the 'sqr' directory may be consolidated into one or more output strips of terrain. These strips span one degree of latitude and one or more degrees of longitude and are written to the 'dma' directory as binary output files. The DMA stage requires use of the boot file created when Suppressor was installed on the system.

Input Files	Output Files
aew/dma/dma_aew.dat	aew/dma/degree_strip.dma[3]
aew/sqr/degree_square.dtl[4]	aew/dma/dma_aew.lis
lan/boot.bin	

EDB

The EDB converts one or more data files covering a parallel of latitude each into a single terrain file output into the 'edb' directory as an 'untranslated' binary terrain file.

Input Files	Output Files
aew/edb/edb_aew.dat	aew/edb/untranslated_aew.edb
aew/dma/degree_strip.dma[5]	aew/edb/edb_aew.lis
lan/boot.bin	

SDB

The SDB is used to initialize the data specific to each modelled scenario. It might require terrain files, which could be either untranslated and located in the 'edb' directory or translated and found in the 'trn' directory. When untranslated data is read in the SDB will output a new translated binary file to the 'trn' directory. The SDB should contain the instruction `DO-NOT-USE-TERRAIN` if no terrain files are to be used at all.

[3] More than one degree wide strip of data may be produced.

[4] There may be several of these files, each constituting a single square degree of DMA level 1 DTED.

[5] There may be several of these files, each constituting a single parallel of terrain data.

As a final complication the SDB may be constructed in several steps with each output SDB binary file being re-read along with a new SDB instruction set to construct the complete SDB. The only significant complication this entails is that all of the SDB files bar the first would use the output translated terrain of the initial SDB step. Hence these files would include the keyword `USE-TRANSLATED-TERRAIN` rather than the `TRANSLATE-TERRAIN` keyword.

Input Files	Output Files
aew/sdb/sdb_aew.dat	aew/sdb/sdb_aew.bin
aew/tdb/tdb_aew.bin	aew/sdb/sdb_aew.lis
aew/edb/untranslated_aew.edb[6]	aew/trn/translated_aew.trn[7]
aew/trn/translated_aew.trn[8]	

MOD

The MOD stage is responsible for model execution and has a relatively uncomplicated set of file dependencies:

Input Files	Output Files
aew/mod/mod_aew.dat	aew/mod/mod_aew.bin
aew/sdb/sdb_aew.bin	aew/mod/mod_aew.lis[9]
aew/trn/translated_aew.trn	aew/sit/sit_aew.bin

When translated terrain information from the SDB is used a second binary output file, (the situation file), is written to the 'sit' directory for possible post-processing. The output listing file, 'mod_aew.lis', is the time history list which describes the simulation's results.

ADB

The ADB may be used to generate a summarized view of the Time History listing. It is unique in that in produces no binary output files, since there are no subsequent phases in the running of Suppressor.

Input Files	Output Files
aew/adb/adb_aew.dat	aew/adb/adb_aew.lis
aew/mod/mod_aew.bin	
aew/sit/sit_aew.bin	

[6] Used only when the SDB processes untranslated terrain files, see the SDB item TRANSLATE-TERRAIN.
[7] Produced when the SDB translates terrain using TRANSLATE-TERRAIN.
[8] Used only when the SDB processes translated terrain files, see the SDB item USE-TRANSLATED-TERRAIN.
[9] This is the 'Time History Listing' used to record the results of the scenario.

The above information can be used to deduce file dependencies, i.e. which input and output files each output file directly and indirectly depends upon. For example, the untranslated binary terrain file 'untranslated_aew.edb' is produced in the EDB step, and clearly depends on the EDB input files: 'edb_aew.dat', 'degree_strip.dma', and 'lan/boot.bin'. The last of these files is the initialization file, and depends on no further files, whilst the first file is a user prepared input file and so also depends on no further files. However the terrain file 'degree_strip.dma' depends on those files used to create it in the DMA step and so on. This analysis is very useful for automatically maintaining the whole model using the Unix 'make' facility which uses information on file dependencies to ensure that all components of a model are current. If a single file were edited or deleted 'make' can be used to automatically recover the effected binary files without the user needing to worry about which files depend on which others.

2.4 Suppressor Input File Format

Although the instruction sets for each of the databases are different, there are some commands that are common to each set. The first, and most important of these, is the `EXECUTE` command, which instructs Suppressor on how to interpret the contents of each file. The `EXECUTE` keyword must be the first word in the file with no characters preceding it other than spaces. The remainder of the line is ignored and on the next line the command `INSTRUCTIONS-FOR:` must appear, which is followed by a label which identifies the current data set. In most cases the label identifying the instruction set is the three letter abbreviation used previously in referring to the stages, but in the case of the UAN and the MOD it is different; the complete set of identifiers being:

Instruction Set	Suppressor Keyword
UAN	`UAN-DEFINITION`
TDB	`TDB`
DMA	`DMA`
EDB	`EDB`
SDB	`SDB`
MOD	`MODEL EXECUTION`
ADB	`ADB`

Following the identification of the instruction set the instructions specific to each data-set follow until they are terminated by an `END-INSTRUCTIONS` keyword. This final command may be used to determine if an output binary file is created. Thus the format for the `EXECUTE` command is:

```
EXECUTE
INSTRUCTIONS-FOR:
<data-set label>
  <instruction-set> ...
END-INSTRUCTIONS <data-option>
```

Here `<instruction-set>` refers to the possible user provided commands, drawn from one of the seven listed sets above, which will be identified by one of the listed labels. The final item, `<data-option>` is one of either `SAVE-DATA` or `DO-NOT-SAVE-DATA` according to whether or not an output binary file is required or not. For example we could have:

```
EXECUTE
INSTRUCTIONS-FOR:
TDB
  <TDB instruction-set> ...
END-INSTRUCTIONS SAVE-DATA
```

The relocatable binary output files, (i.e. the '.bin' files), of the terrain processing steps are not required for subsequent stages in the process, so for the DMA and EDB data-sets the `DO-NOT-SAVE-DATA` option may be used, so that for the DMA we have,

```
EXECUTE
INSTRUCTIONS-FOR:
DMA
  <DMA instruction-set> ...
END-INSTRUCTIONS DO-NOT-SAVE-DATA
```

The instructions included in each data-set may be include comments which use one of two forms, the first is for single line comments using a dollar symbol `$`, for example:

```
$ This is a single comment line in Suppressor
```

Comments which span multiple lines can be introduced by using matched pairs of the symbols `$>` and `$<`. Thus a multiple line comment can be given as:

```
$> AOD AIRBORNE EARLY WARNING SCENARIO

  This file contains definitions for the
  AOD Airborne Early Warning Scenario.

$<
```

All Suppressor input lines, including comments, must not contain more than 72 characters. This is most probably an anachronistic survival from the days when computer input was prepared using punched cards, but care must be taken that the Suppressor input files respect this. The processing steps will not explicitly warn the user of this problem but lines will be silently truncated. (This eventuality can be detected using third party software or by careful examination of the output listing files, which echo Suppressor's interpretation of the command lines.)

2.5 Summary

The overall design of the Suppressor model has been discussed with emphasis placed on the role of the SAIL and the internal model of mental processing. The primary function of the thinking process in the evaluation of the scenario should be noted, along with the implications this holds modelling complex scenarios. In addition the relationship between the various data files both read and written by Suppressor, and where they may be found in the AOD implementation has been described. The subsequent sections of this guide now concentrate on the preparation of these individual data-sets. The guide's role here is to describe the many different ways a particular system may be modelled using Suppressor. It is certainly not intended that the examples shown be seen as the 'only' or even the 'best' way to encode a particular situation into a Suppressor scenario. Indeed it would be surprising if there were a single best route. Usually it would be expected that different approaches will prove suitable for different scenarios.

3. Constructing the Type Data Base

The Type Data Base (TDB) is the repository of all the information concerning the player-types to be used within a Suppressor scenario. Suppressor has no built in knowledge of player-types or of their constituent systems so all these data must be provided by the user. This is done using the TDB instruction set, the description of which forms the bulk of this section of the guide.

As noted previously Suppressor processes the user's TDB file immediately following its reading of the 'User Application Names' (UAN) data, which list all the names of all the user defined entities within the scenario. Suppressor allows that the UAN be included within the same file as the TDB data and processed in a single step; following this practice we shall treat these two databases as a single entity. It should be noted that the UAN also contains the names of items which do not appear until the SDB, for example, the names of the scenario's sides and the checkpoints used in the scenario. Since more than one scenario can use a single TDB this may mean that some names listed in the UAN will not feature in all of the scenarios. This is not a serious problem, since although everything used in a scenario or defined in the TDB must be uniquely named, not every name given in the UAN must be used.

The first part of this section discusses the format of the UAN, then the TDB is described in detail. Special attention is given to the tactics which must be defined in the TDB for each active player-type since these are the single most important factor in determining how a Suppressor simulation will develop.

3.1 User Application Names

Every Suppressor model must commence with the UAN, which as mentioned previously lists the names of all the user defined items in the scenario. It may either be present in its own file or processed as part of the TDB stage when it is included as the first portion of a TDB input file. Although the TDB itself may be built up from many separate files only one UAN block may occur and this is at the beginning of the first TDB file in the processing sequence.

The UAN instruction set starts with the keyword `UAN-DEFINITION` followed by a list of UAN categories. No category can be repeated and each lists the names of the user declared items which belong to it. There is no limit to the number of names that may be listed, although each entity's name must be unique. By convention user names are distinguished from Suppressor keywords, which cannot of course be used to name user defined items, by typing them in lower case, although upper case names may be used if desired. The listing of category lines is terminated by an `END-UAN` command. The format of the UAN definition is thus:

```
$ A fragment of a User Application Names Database
$ As an aid to clarity keywords are shown in bold
UAN-DEFINITION
$ The category SIDES names the conflict's principals.
$ There may be more than two sides if it is desired.
  SIDES
    Blue          Orange

$ A few of the player-types used in the conflict are
$ given here. Player's so named may figure for any side
  PLAYERS
    aew_player    bomber_player
    sam_player    fighter_player
..<Omitted name declarations>
END-UAN
```

In the above example Suppressor keywords are printed in bold type, this convention will be maintained throughout the main portion of this guide. Text indicating that portions of an instruction set have been omitted is enclosed in angle brackets, for example '`<Omitted name declarations>`'. A fixed width font is used to better represent the appearance and indentation of a real data file.

Altogether there are twenty-three categories in the UAN which can be divided into specific groups. However, not all of these categories may be required in a particular scenario, in which case they may be omitted from the UAN. There is no specific order in which these categories must be introduced in the UAN. In the following we shall first introduce the categories that are to do with player-types, their physical and mental components followed by their associated capabilities and resources. Finally the categories that are associated with organization of players, their command and control and the communications between them.

The first categories we shall discuss are the two which together represent the players used in the scenario: *players* and *elements*.

SPECIFIC PLAYERS	
UAN-CATEGORY	**Category Description**
PLAYERS	The names of the types of players in the scenario.
ELEMENTS	Each player is composed of one or more elements with different locations and properties. All elements must be listed.

It should be now explained that players in Suppressor can have a very flexible structure. Players do not have to be single or even in localized structures, for example, an AEW craft, a fighter aircraft or a cruiser. Players can in fact consist of many elements at multiple locations. An example is a single player which represents a Surface-to-Air missile defence system. This player could have multiple batteries of SAM missiles at several sites, with a command post and long, medium and even short range radars at other sites. Each of these separate components will be represented by Suppressor as an 'element' of the whole player, each element being in turn composed of constituent 'systems'. It is these systems that are explicitly associated with the generic functions that are the fundamental building blocks of Suppressor.

The generic functions, we recall, may be divided into five active or physical functions: *sensing, talking, shooting, disrupting,* and *moving,* and the mental service function of *thinking*. This latter function may be subdivided into *noticing, digesting* and *reacting*. This subdivision is only really apparent to the user when defining the capabilities of each thinker to process its player's tactical plans. Each of these six functions has a specific physical system associated with it, but note further that in practice sensing and talking are both divided into transmission and reception. The systems form the basic building blocks used to describe the player. The eight UAN categories that correspond to these six generic functions are as follows:

FUNCTIONAL SYSTEMS		
UAN-CATEGORY	Category Description	Generic Functions
MOVERS	Movers, as the name suggest, may move during the scenario and so change their location. Examples of movers could be aircraft, ships or motor vehicles.	MOVE
WEAPONS	Weapons allow players to attack other players	SHOOT
THINKERS	Thinkers are able to make decisions based on the information they receive from the model Typical thinkers would be pilots commanders or even computer systems.	THINK
DISRUPTORS	Disruptors, which impair other players' abilities to TALK and SENSE are declared here.	DISRUPT
COMM-RECEIVERS	Communication receivers allow a player to receive messages.	TALK
COMM-TRANSMITTERS	Communication transmitters enable a player to send messages. Players with both transmitters and receivers may talk with other players.	TALK
SENSOR-RECEIVERS	Sensor receivers allow the detection of other players.	SENSE
SENSOR-TRANSMITTERS	Sensor transmitters are used with radar sensor receivers in order to detect other players.	SENSE

The physical systems described above have certain attributes in addition to their generic functionality. In particular, they may consume the explicitly modelled resources of fuel or ordnance. They also have certain *capabilities* which describe the physical characteristics of the system and *susceptibilities* which describe the player's interaction with the environment, for example. its radar cross section. As a separate category the frequency bands in which an infrared sensor receiver may operate must also be named.

RESOURCES AND CAPABILITIES	
UAN-CATEGORY	**Category Description**
FUEL	Fuel is a continuously expended resource employed by movers.
ORDNANCE	Ordnance is used by weapon systems and may, for example, consist of missiles or bombs
FUTURE PLAYERS	Both weapon systems and thinkers can spawn new players, known as future players. An example would be a missile battery which launches a guided missile; this missile is now a full player with its own capabilities, tactics and susceptibilities.
CAPABILITIES	Each element defined in the TDB has its own capabilities and limitations. For example aircraft have maximum and minimum operating speeds, weapons have maximum rates of fire and ranges. Generic capabilities for some systems, (such as an antenna pattern), may be defined and named and then shared between many separate systems in the TDB.
IR-BANDS	The list of all the infrared frequency bands an infrared sensor receiver may operate in. These bands do not have to have their frequency ranges explicitly set, but the amount of radiation and reflectance of a target will depend on which band it is being observed.
SUSCEPTIBILITIES	If a player is to be detectable by another player in the scenario, data describing the coupling between these components are needed. For example, if an aircraft is to be seen by some radar system tables giving the aircraft's radar cross-section data are needed. If such tables are named here then they may be re-used by more than one player in the system as a generic susceptibility. A player with no susceptibilities is undetectable.

A characteristic feature of a single player in a Suppressor scenario is that it has a single set of *tactics* which are shared by all the elements of the player. Furthermore if one or more of these elements are movers they potentially share the same movement plans, known in Suppressor as *maneuvers.*

PLANS AND TACTICS	
UAN-CATEGORY	**Category Description**
TACTICS	Tactics are employed by players during model execution. Each player may have a combination of its own private tactics and generic tactical plans shared by other players in the scenario.
MANEUVERS	Plans which describe movement, such as pre-planned aircraft course or the action to take near a threat or target should be named here.

All items that are to do with movement, not just names of genuine manoeuvres but also names of checkpoints on pre-programmed paths and other entities, are called . maneuvers.

When, as described above, many elements are used to model a single player in Suppressor all these elements share a single set of tactical plans and are assumed to be in perfect communication with each other. This representation is an example of a notional detail or low fidelity model. Higher fidelity (fine detail) may be achieved by splitting the player up into several different players in a single chain of command. Each of these new players will have its own set of tactics and plans and will now need to have explicitly modelled communications with each other.

An example of varying the resolution of a model is given by the problem of representing a wing of fighter aircraft. These can either be slaved together under a single set of tactics as a single player, or linked together in a command chain as a set of independent players each with their own tactics. In general the latter description, with its finer detail, will give a higher degree of fidelity in modelling the military situation. It will also impose greater demands on the computer system used in running the scenarios and on the work required in generating the model in the first place. The user must decide in designing the scenario how accurate a representation to use by taking these considerations into account.

Several user defined items must be provided to describe the organization, command and control of and communication between distinct players. Most of these items will be found within the SDB, although they may be referenced within the TDB in some places. Five categories exist which contain such entities: *sides, command chains, zones,* and communication networks which are either *explicit nets* or *implicit nets.*

COMPOSITE SYSTEMS	
UAN-CATEGORY	Category Description
SIDES	The names of the sides in the conflict
COMMAND CHAINS	Command and control structures defined in the scenario are categorized as command chains in Suppressor. Each chain must be declared in the UAN.
ZONES	Each command, control and co-ordination area, e.g. the region wherein an AEWC aircraft reports to a central commander, is designated as a zone.
EXPLICIT NETS	Communication nets that may be jammed or be masked by the terrain or other causes are classified as explicit nets. In this case computations will be made at run time to check that the link is available.
IMPLICIT NETS	Communication nets that cannot be jammed or are not subject to terrain masking, are classified as implicit nets. Here run time checks on availability are not needed.

The sides in a scenario represent the conflicting parties, there must be one or more side in a single scenario. Command chains represent the (various) command structures which are used to control the players which make up each sides forces. A single player may figure in several command chains. (Notice that the relationship between players and command chains is much more flexible than that between elements and players, a single element cannot of course appear in more than one player.) Many command and control zones referring to specific spatial regions of the combat area can be defined: reporting, engagement and command responsibilities will typically vary for players as they move from one zone to another. The communication nets the players use to talk to one another with may be explicitly modelled as broadcast radio frequency messages in explicit nets, or modelled as implicit nets using landlines or other secure means.

The remainder of this section concentrates on the definition of the TDB, all of whose entities must have been named in the UAN. In practice the UAN and TDB are likely to be assembled incrementally, with items added to the UAN as they become needed in the TDB. No items needed exclusively for the SDB need be named until an SDB file is in preparation, although no harm will come if these are anticipated. (Like, for example the names of the sides in the scenario.)

3.2 Type Data Base Instructions

The TDB instruction set is by far the most important set of data used in defining a Suppressor scenario. These data describe all the physical characteristics and capabilities of the modelled systems and their interactions. The data included here do not just list things such as maximum speeds and radar power levels but also the tactics and capabilities of the player's thinkers. These are crucial in determining how players respond to physical stimuli within each scenario. The tactics in fact determine the outcomes of the mental processing activities that were described in section 2. There it was described how each thinker first notices a stimulus, then digests it to update its perceptions of the outside world, and then reacts to these perceptions as appropriate. The results of this final 'reaction' stage are determined by each player's tactical blocks. These will describe whether or not a player should attack or flee or just do nothing when faced with the current circumstances.

Processing the TDB instruction set occurs either as the second phase of a Suppressor model run, i.e. immediately following the processing of the UAN data set, or as the second part of a combined UAN and TDB phase. The TDB instructions themselves, both if they occur in their own data file or share a file with the UAN block, are identified with the keyword `TDB` followed by a mandatory comment. This comment may be one or more lines long, but must always be present. These comment lines need not be preceded by the normal comment identifier, (a '$' symbol), since Suppressor always interprets the lines following the `TDB` keyword as comments. The comments are terminated by either the keyword `INSERT-MODE`, (used in the TDB instruction itself to generate new TDB instructions), or the keyword `REPLACE-MODE`, (used only in the SDB processing phase to over-ride specified TDB data structures for a single scenario). The format of the TDB definition in the TDB instruction set is thus:

```
TDB
The TDB instruction always requires one or
more comment lines after the TDB keyword.
INSERT-MODE:
    <TDB instructions>
END-TDB
```

The TDB, once complete, forms a data base which is available for many different scenarios. We may therefore see the TDB as a repository of functional units, (consisting of players and their physical capabilities, plans of engagement and movement plans), which may be re-used in the same or modified form in many different studies using the Suppressor. It is this great flexibility which is at once Suppressor's greatest strength and the cause of the greatest difficulty in successfully using Suppressor.

The TDB instructions can be broken down into four functional categories which form the basic structure of a TDB file, these being:

- **PLAYER-STRUCTURE**, each player-type mentioned in the SDB, i.e. used in the scenario, will be associated with a `PLAYER-STRUCTURE` block in the TDB. Its role is to be a blueprint for the player-type; multiple instances of these types may then be used in the SDB without having to re-specify all the players' properties on each occasion.
- **CAPABILITY**, player-types consist of sub-systems which have a particular generic function associated with them, (thinking, shooting, moving etc.). The `CAPABILITY` block is used to completely specify each of these systems' physical characteristics. In the case of a mover relevant entries will include maximum and minimum speeds and acceleration, limiting rates of turn, maximum rates of ascent and descent and so on.
- **SUSCEPTIBILITY**, all player-types, in interacting with the environment and the other player-types, betray certain susceptibilities. These define the electro-magnetic emissivity and reflectivity of the player-type, which in turn determine how likely a player is to be detected by a sensing system, (such as a radar), in any given situation.
- **TACTIC**, it has already been noted that players are characterized by their tactical abilities, which are defined in the `TACTIC` block. Several player-types may share the same tactics, or each tactical building block may be unique to a particular player-type. Tactics itemize the player-type's manoeuvres and plans of engagement, intelligence reporting and strategic goals.

These four categories will now be discussed and their constituent interactions described.

3.3 The Player Structure

The PLAYER-STRUCTURE building block is used to introduce each player-type required in the scenario. It contains no data entries itself but instead identifies which TACTIC, SUSCEPTIBILITY and CAPABILITY building blocks go together to define a particular player-type.

```
$ Portion of a PLAYER-STRUCTURE for an AEW craft
PLAYER-STRUCTURE aew_player
  TACTIC   aew_tactics[10]
  LOCATION 1[11]
    ELEMENT       11 aew_craft   DISCRETE 1
      <Definitions for AEW player>
END PLAYER-STRUCTURE
```

The above display shows part of an example PLAYER-STRUCTURE definition for an AEW craft. Items in a bold font are keywords and comments are preceded by the '$' character while numbered superscripts are for annotation only. The first line of the definition names the player-type concerned, in this case the aew player-type. It is followed immediately by the name of the tactical block used by this player-type. Since each player uses one set of tactics for all its constituent elements only one named TACTIC block is used within a player-structure. Furthermore tactics *must* be defined for all active players (i.e. player-types which are able to process and respond to events occurring in the scenario).

[10] The named tactical building block (aew-tactics) must be included elsewhere in the TDB.

[11] A player-type may be distributed across several physical locations, and so even when only one element is present each element must have a uniquely numbered LOCATION associated with it.

Below the tactical definition occur one or more LOCATION statements for the player-type. In our example above, the player-type represents a single aircraft so a single location is all that is required. At each location one or more elements can be defined, these elements are composite entities which will usually have some readily identifiable military role, or represent a specific piece of military equipment. In this case a single ELEMENT, representing the AEW craft itself, is all that is required, and all that is legal since the element represents an entry which can move. In many situations multiple location and element statements are appropriate. For example, consider a player-type representing a surface-to-air missile (SAM) battery:

```
$ A portion of a SAM missile system player
PLAYER-STRUCTURE sam_player
  TACTIC   sam_tactics
  LOCATION  1
    ELEMENT 10 sam_hq      DISCRETE 1 CRITICAL
      <Definitions for the SAM headquarters>
    ELEMENT 11 sam_radar   DISCRETE 1 CRITICAL
      <Definitions for the SAM radar sensors>
  LOCATION  2
    ELEMENT 20 sam_battery DISCRETE 1
      <Definitions for the 1st SAM battery>
  LOCATION  3
    ELEMENT 30 sam_battery DISCRETE 1
      <Definitions for the 2nd SAM battery>
END PLAYER-STRUCTURE
```

Here multiple elements occur at several locations, but they constitute a single player with a single set of tactics. The first numbered location statement is associated with a headquarters element. This element would contain systems such as communication devices along with thinkers representing commanders. It controls two distinct batteries of missiles, each represented by distinct elements which may be situated in different places and a radar system located with the headquarters element. Note that the physical location of these elements is not set in the TDB, but rather in the scenario data base, (SDB). This implies that the physical location of two distinct logical location identifiers may in fact be identical. By modelling the SAM system as a single player implies that all communications between different components of the system are perfect.

It is clear from the above example that both the LOCATION and ELEMENT keywords are identified with numerical labels. The role of these labels is to provide a unique identification for each entry both within the PLAYER-STRUCTURE template and within the scenario as a whole. For example, the SAM site player-type has two otherwise identical 'sam_battery' elements distinguished by the labels '20 sam_battery' and '30 sam_battery'. These full specifications must be unique within the whole scenario whilst the labels alone, i.e. 20 and 30, need only to be unique to the player-type itself.

The convention used in this guide is that the locations are numbered in ascending order starting with one. The elements modify the label of their location by appending their number, in this case starting at zero. (Hence the second element at the third location will have label 31.) Since no examples are offered with more than nine locations, or ten elements per location, this generates unique labels.

Some player-types may be required which cannot be completely destroyed, (a passive player-type which functions as a target for example). In this case we may have:

```
$ A complete specification of a passive target
$ (here identified as a bridge)
PLAYER-STRUCTURE target
  LOCATION  1
    ELEMENT 10 bridge        CONTINUOUS 1.0
      SUSCEPTIBILITY bridge_signature
END PLAYER-STRUCTURE
```

In this example the element's nature is `CONTINUOUS` with the real value of one indicating the element's initial survival probability. This value will reduce as it is attacked but will never diminish to zero. Note that since this player-type performs no actions and processes no stimuli it does not need a `TACTIC` block nor any corresponding thinkers, nor indeed any physical systems at all. One extra new instruction is included here: before a target such as the bridge can be attacked it must be perceived by an opposing player. For this to happen the bridge must first be 'sensed' by this player's sensor systems. These sensor systems will only be able to detect the target if the target has an associated electromagnetic 'signature' or cross-section defined. This is done in a `SUSCEPTIBILITY` block, and in the target's player-structure this block is identified.

In the case of elements with a `DISCRETE` nature the initial value is an integer indicating the number of 'kills' required to destroy a target. For example, the following player-type definition:

```
$ Define cat and dog as player-types
PLAYER-STRUCTURE animal
  LOCATION  1
    ELEMENT 10 cat        DISCRETE 9
      <cat characteristics>
    ELEMENT 11 dog        DISCRETE 1
      <dog characteristics>
END PLAYER-STRUCTURE
```

indicates that a cat has nine lives, but a dog only one.

The difference between the discrete and continuous cases is as follows:

- `DISCRETE`, integer valued quantity: a random number is selected when the element is attacked, the probability of a successful attack. If this number is less than some user defined threshold, (which varies according to the circumstances of the attack and the weapon used and is known as the probability of kill or P_K), then a 'kill' is recorded and the value decremented by unity. When this value reaches zero the element is completely destroyed. Otherwise it is considered to survive and remain functional.
- `CONTINUOUS`, real valued quantity: this represents the element's cumulative probability of survival. After each new attack this probability is multiplied by the complement of P_K, which is the probability of the element surviving the attack. Note that while this value may become very small it will not reach zero and so players of this type will always remain within the scenario.

The SAM missile battery's first location contained the following elements:

```
LOCATION  1
  ELEMENT 10 sam_hq      DISCRETE 1 CRITICAL
    <Definitions for the SAM headquarters>
  ELEMENT 11 sam_radar   DISCRETE 1 CRITICAL
    <Definitions for the SAM radar sensors>
```

We see that the final entry on the element's definition line is that the SAM headquarters element and the SAM radar element at the first location are both `CRITICAL`. This keyword means that the destruction of either element would destroy the other element at the same location. If this keyword is absent it is equivalent to using the keyword `NONCRITICAL`, the default value, which would allow one element to survive the destruction of the other. A continuous element is always treated as being `NONCRITICAL`, since it can never be completely destroyed. The SAM missile batteries would survive the destruction of the `CRITICAL` elements but may no longer be operational if they do not have local tracking and targeting capabilities.

Let us return to the example of the AEW `PLAYER-STRUCTURE` building block, this time extending the definition to include both the player-type's signature and a physical system, its mover:

```
$ Example of a PLAYER-STRUCTURE for an AEW craft
PLAYER-STRUCTURE aew_player
  TACTIC   aew_tactics
  LOCATION 1
    ELEMENT        10  aew_craft DISCRETE 1
    SUSCEPTIBILITY     aew_signature
    MOVER        1010  aew_airframe
      CAPABILITY       aew_airframe_data
      FUEL        jp4  CONTINUOUS 4000.0 (KG)
    <other systems>
END PLAYER-STRUCTURE
```

The mover is labelled and named in much the same way as the element of which it is a part. Again the numerical label must be unique within the current player-type, and when combined with the name of the mover 'aew_airframe' provide a unique identification of the system. (There cannot be another '1010 aew_airframe' belonging to any other player-type, although there may be a '1010 tank'.) There is no required order for the specification of the systems, they may be introduced in any sequence.

Just as with location and element labelling this guide uses a convention to generate system labels. Each system is labelled by appending two digits to its element's label. The first extra digit identifies the generic function of the system and the second digit is incremented from zero to differentiate between systems of the same type.

System Type	Code Digit
Thinker	0
Mover	1
Communication Device	2
Sensor	3
Weapon	4
Disruptor	5

In counting the above systems, receivers always have even labels, starting with 0, and transmitters always have odd labels, starting with 1. The above scheme will give unique four digit labels provided no more than ten physical systems of each type are defined within each element. (The limit being five each for transmitters and receivers.) In practice this is quite generous enough for this guide, in cases where more components are required more than one digit may be used in each stage of assembling the label.

In addition to naming each system the system's physical capabilities must also be identified. The `CAPABILITY` instruction accomplishes this by naming a data block containing the relevant physical data. The `CAPABILITY` identifier can be used quite flexibly in order to complete the definition of all the physical characteristics of a system. Each system may have its own individual `CAPABILITY` block; or several systems may have identical capabilities and share a single block; or a system's total capabilities may even be built up by using several `CAPABILITY` blocks. An example of the latter case is where a block of basic capabilities is shared by many similar systems with their peculiar features defined in separate `CAPABILITY` blocks.

The AEW craft has the ability to move, sense, and talk and so consequently to think. It requires specific functional systems to do these tasks, with the mover being the first of these. Considering these systems in turn:

- **MOVER**: moving.

```
MOVER          1010 aew_airframe
  CAPABILITY        aew_airframe_data
  FUEL         jp4  CONTINUOUS 4000.0 (KG)
```

Movers may use the consumable 'FUEL'. If so this is specified following the mover's capability blocks. The fuel used must be identified, here 'jp4', although the name chosen is immaterial. Fuel is measured as a continuous quantity and this is specified along with the amount of fuel initially available, here 4000 kg. The mover's CAPABILITY block will include a FUEL-USAGE entry which specifies how much fuel the mover burns at different speeds and altitudes, once the fuel is exhausted the mover will crash and will be regarded by Suppressor as being destroyed. No residual glide or coast capability is automatically modelled by Suppressor[12]. A mover may function without using any fuel at all if the burn rate specified in the FUEL-USAGE instruction is zero, or if either of the FUEL-USAGE or FUEL entries are missing.

- **THINKER**: thinking.

```
THINKER        1000 aew_commander
  CAPABILITY        aew_commander_think_data
  FUTURE-PLAYER     interceptor DISCRETE 3 (COPIES)
THINKER        1001 aew_operator
  CAPABILITY        aew_operator_think_data
```

This example includes two separate thinkers each with their own distinct capabilities. They will share the same TACTIC block 'aew_tactics' and which will be completely aware of the other's perceptions at all times. Each thinker will process different events, as discussed in the overview of Suppressor's modelling of mental processing in section 2.1. The two thinkers can be thought of as explicitly modelling two crew members, or as simply a way of dividing up the responsibilities of the crew between two separate thinkers. In general, the more thinkers that are provided for a player the more information it will be able to process before being overwhelmed with data. The number, and role, of thinkers provided for each player-type is an important aspect of modelling in Suppressor and should be considered with care by the user. The commander can also use a resource, 'interceptor' which corresponds to a entirely new player which may be brought into action as a subordinate of the AEW commander. This subordinate's player-type will have a template defined in the TDB whose name is identified in the thinker's capability block, 'aew_commander_think_data'.

[12] There are several ways of enabling a mover to have a residual glide or coast ability by using either the FUEL-USAGE or the FUTURE-PLAYER instructions.

- **COMM-RCVR**: talking.

```
COMM-RCVR       1020 aew_radio_rx
  CAPABILITY         radio_rx_data
  CAPABILITY         aew_radio_rx_data
```

Used to define a communication receiver. Here the receiver has physical capabilities defined with the help of two capability blocks, the first giving the characteristics of a generic radio receiver 'radio_rx_data' and the second 'aew_radio_rx_data' specifying some peculiarities of the AEW craft's radio. Although talking is a single generic function the receiver confers only the ability to *listen*, to *speak* a transmitter is also required.

- **COMM-XMTR**: talking.

```
COMM-XMTR       1021 radio_tx
  CAPABILITY         radio_tx_data
```

A communication transmitter, which allows a player to speak is defined with this entry. A (possibly) generic radio transmitter is used in the above example, 'radio_tx'. With a radio receiver and transmitter the AEW craft can both send and receive messages but, for proper operation, these two systems need to be 'linked', which will be explained below.

- **SNR-RCVR**: sensing.

```
SNR-RCVR        1030 aew_radar_rx
  CAPABILITY       aew_radar_rx_data
```

Suppressor can model four kinds of sensing using radars, radar warning receivers, which listen in to other radar systems, optical sensors and infrared sensors. The AEW craft uses a radar system for early detection so it requires both a receiver and a linked transmitter.

- **SNR-XMTR**: sensing.

```
SNR-XMTR        1031 aew_radar_tx
  CAPABILITY       aew_radar_tx_data
```

Only radars, which are active sensing systems, need a sensor transmitter which must be explicitly 'linked' to its corresponding receiver.

The final portion of the player-type template is used to define some explicit linkages between systems. Most systems can interact properly using the mental processing model of Suppressor without needing to be explicitly linked. There are however four exceptions to this rule and these systems must be explicitly linked:

- communication receivers with communication transmitters,
- sensor receivers with sensor transmitters,
- weapons with tracking sensor receivers,
- thinkers with sensor receivers.

In all these cases the role of the link is to ensure that events are processed by the appropriate combination of systems. For example our AEW craft has two thinkers whereas its radar should be monitored by only one of them, who represents the radar's operator. By explicitly linking one thinker, the operator, with the radar it can be ensured that all the sensing events from this receiver, (such as the detections that are stored in the player's iconic memory), can only be noticed by the operator. If this linkage is not provided no detections will ever be noticed and the AEW craft would be effectively blind.

The AEW craft in the example has no weapons so three explicit linkages need to be made with the following format:

```
LINKAGES
  1001 WITH 1030  1020 WITH 1021  1030 WITH 1031
```

The systems are identified solely by their labels and the ordering of the systems in the linkage phrase is inconsequential. The `LINKAGES` portion of the `PLAYER-STRUCTURE` is the last element of the template, and so with its completion the whole player-type is defined.

We have yet to consider the systems which correspond to the generic functions of shooting and disrupting. Any player-type which can lethally engage, i.e. attack, another can shoot and needs to carry a weapon. Disrupting is the ability to non-lethally engage another player, i.e. to interfere with this player's performance without destroying or damaging it. In Suppressor disruption works by degrading the ability of an opposition player to perform its generic functions of sensing or talking, and this is accomplished by the disruptor targeting the player's sensor and communication *systems*, not the player itself. Electronic counter measures (ECM) are a typical example of disruption, and Suppressor can explicitly model the performance of electronic jamming. Examples of systems of these types follow below:

- **WEAPON**: shooting.

```
WEAPON          1040 missile_launcher
  CAPABILITY         missile_launcher_data
  ORDNANCE           missile DISCRETE 6 (ROUNDS)
```

The above weapon is a missile launcher which fires a consumable, (ORDNANCE). Unlike fuel for a mover this is a DISCRETE quantity of which a fixed number of rounds are available: here a total of six missiles, known as ROUNDS (of ammunition).

- **DISRUPTOR**: disrupting.

```
DISRUPTOR       1050 ecm_jammer
  CAPABILITY         ecm_jammer_data
DISRUPTOR       1051 flare_launcher
  CAPABILITY         flare_launcher_data
  CM-EXPENDABLE      flare DISCRETE 40 (NO-UNITS)
```

The above jammers might be used to disrupt enemy systems like sensor receivers and communication receivers. Furthermore the effectiveness of weapon systems can be degraded too because these systems rely on tracking sensors to select their targets. Of the two disruptors one, '1051 flare_launcher' uses expendable countermeasures, a resource identified with the CM-EXPENDABLE label which, like ordnance, has a discrete nature. In the example the launcher is stocked with 40 flares.

The weapon system '1040 missile_launcher' referred to above has only the capability to fire missiles as 'dumb' ORDNANCE, which is modelled relatively crudely by Suppressor. If the missiles being modelled are complex actively guided types, carrying their own sensors and communication equipment, along with a computerized 'thinker' which can take decisions, then a better model is required. This can be accomplished by modelling the missile as a player, with its own PLAYER-STRUCTURE block, tactics and physical capabilities. However, unlike most other players modelled in a scenario this player will only come into existence when fired by the weapon. Because such players are created dynamically by the model at run time and not initialized by the user they are known as 'future players'. An example of using a FUTURE-PLAYER to model a missile is provided by the SAM missile battery introduced earlier:

```
$ Portion of a SAM site PLAYER-STRUCTURE definition
PLAYER-STRUCTURE sam_player
<Definitions for the SAM site player>
  LOCATION 3
    ELEMENT       30   sam_battery DISCRETE 1
      SUSCEPTIBILITY  sam_signature
      WEAPON    3040   sam_launcher
        CAPABILITY     sam_launcher_data
        ORDNANCE       sam_msl DISCRETE 18 (ROUNDS)
        FUTURE-PLAYER sam_fly DISCRETE  6 (COPIES)
<Linkages for the SAM site player>
END PLAYER-STRUCTURE
```

Two kinds of missile may be launched by this weapon: first a missile modelled as `ORDNANCE` using notional detail with name 'sam_msl'. The launcher initially has 18 of these missiles. Secondly, six missiles may be launched using the 'sam_fly' `FUTURE-PLAYER` resource. This feature allows a completely new player to be created by the sam player-type. The name of the player-type that will be created is identified in the `CAPABILITY` block 'sam_launcher_data' as being a 'sam_missile_player'. (This process is described in the `RESOURCE-DISAGGREGATION` entry of the `CAPABILITY` block.)

The 'sam_missile_player' is a fully fledged player-type and all its definitions and capabilities must be included in the TDB in the normal way. The example below gives a possible `PLAYER-STRUCTURE` block for this player-type. Note that the missile must carry its own weapon in order to carry out its role. In this case it is just a warhead, but it could even be another `FUTURE-PLAYER`. Ultimately, however, some player-type in the sequence must simply deliver 'ordnance' to its target.

```
$  A complete specification of a SAM missile player
PLAYER-STRUCTURE sam_missile_player
  TACTIC sam_missile_tactics
  LOCATION 1
    ELEMENT       10   sam_missile DISCRETE 1
      THINKER     1000 sam_missile_thinker
        CAPABILITY     sam_missile_thinker_data
      SNR-RCVR    1030 sam_missile_radar_rx
        CAPABILITY     sam_missile_radar_rx_data
      SNR-XMTR    1031 sam_missile_radar_tx
        CAPABILITY     sam_missile_radar_tx_data
      WEAPON      1040 sam_missile_warhead
        CAPABILITY     sam_missile_warhead_data
        ORDNANCE       explosive DISCRETE  1 (ROUNDS)
      MOVER       1010 sam_missile_body
        CAPABILITY     sam_missile_body_data
        FUEL rocket_fuel CONTINUOUS  200.0 (KG)
      COMM-RCVR   1020 comm_rcvr
        CAPABILITY     comm_rcvr_data
      COMM-XMTR   1021 comm_xmit
        CAPABILITY     comm_xmit_data
  LINKAGES
    1000 WITH 1030  1030 WITH 1040
    1020 WITH 1021  1030 WITH 1031
END PLAYER-STRUCTURE
```

In the above examples, we have seen that some systems can use consumable resources, in particular: mover's may use fuel, weapons fire ordnance or launch future-players, thinkers may also launch future-players and disruptors may use 'expendable counter-measures'.

The complete set of possible uses of consumable resources by systems are:

- Mover systems which use fuel:

```
MOVER        1210 mover_system
  CAPABILITY      mover_system_data
  FUEL            mover_fuel CONTINUOUS 100.0 (KG)
```

- Weapon systems which use ordnance:

```
WEAPON       1040 weapon_system
CAPABILITY        weapon_system_data
  ORDNANCE        bullet   DISCRETE 100 (ROUNDS)
```

- Weapon systems which spin off `FUTURE-PLAYERS` as ordnance:

```
WEAPON       2141 weapon_system
  CAPABILITY      weapon_system_data
  FUTURE-PLAYER missile DISCRETE 6 (COPIES)
```

- Thinker systems which can launch `FUTURE-PLAYERS` as subordinates:

```
THINKER      3000 air_commander
  CAPABILITY      air_commander_data
  FUTURE-PLAYER interceptor DISCRETE 3 (COPIES)
```

- Disruptor systems which eject expendables:

```
DISRUPTOR  1150 disruptor_system
  CAPABILITY    disruptor_system_data
  CM-EXPENDABLE chaff DISCRETE 40 (NO-UNITS)
```

Once the template of the player-type is complete work can begin on its complete physical and tactical characteristics. As noted above these consist of three parts:
- Physical characteristics are given in the `CAPABILITY` block.
- Tactics are given in the `TACTIC` block.
- Interactions with the environment, i.e. radar and other cross-sections, are defined in the `SUSCEPTIBILITY` block.

There are areas of overlap from each of the above data blocks to the others, for example the processing of the tactics by each player-type depends on the capabilities of its thinkers. Bearing this in mind the next three sub-sections discuss these data blocks in detail, commencing with physical capabilities and leaving the crucial tactical block to last.

3.4 The Capability Block

The `CAPABILITY` block of the TDB defines the capabilities of each player-type's functional systems. Each of the `CAPABILITY` blocks named within `PLAYER-STRUCTURE` templates are identified by their user defined name and delimited by the keywords `CAPABILITY` and `END CAPABILITY`. So for example the mover '1010 aew_airframe' was identified with the player-type's template as:

```
MOVER         1010 aew_airframe
  CAPABILITY       aew_airframe_data
  FUEL          jp4 CONTINUOUS 4000.0 (KG)
```

and hence it would have its capabilities defined within the following block:

```
$ An example capability block
CAPABILITY aew_airframe_data
  <capabilities for the aew_airframe>
END CAPABILITY
```

There is not necessarily a one to one correspondence between systems and capability blocks. Each system may use more than one block and each block may be used for several systems.

Since each system is identified with one of the generic functions of thinking, moving, sensing, talking, shooting or disrupting it is convenient to discuss the capabilities relevant to each of these functions in turn. The remainder of this section therefore discusses the capabilities of movers, weapons, thinkers, sensors, disruptors and communication devices in turn. It should be noted that there is some overlap between the capabilities of the systems which transmit or listen in upon radio frequency (RF) emissions, i.e radar sensors, radio communication devices and ECM jammers.

3.4.1 Movers

Mover systems are required for any element of a player-type which is able to move, such as an aircraft, ship or indeed any vehicle. Movement as such is not explicitly modelled by Suppressor, i.e. it has no knowledge of the physical niceties of gravity, force, reaction and momentum. Instead movement is simply encoded as the translation of an entity (i.e. an element of a player at a given location) from one point within the scenario's environment to another. It is largely up to the user to ensure that the movements implied are in fact sensible; Suppressor will happily allow a bomb to fall upwards or a tank to fly if the user requests it.

The mover capability block assists both the user and Suppressor to represent motions realistically. Its major function is to allow limits to be placed on the speed and acceleration of a mover, and to specify how fast the mover burns its fuel, should it use any.

Since a vehicle can be in only one place at any one time the element which represents this vehicle must not have more than one mover system. This restriction does not apply to the other component systems, an element may have several thinkers for example. It is usually convenient to think of the mover as the vehicle's 'body' to emphasize this restriction. A corollary of this is that a moving element should be the only element defined at its `LOCATION` in the player-type's template. This is because each logical location models a physical location and the concept that the elements at a single location are close would break down if they could move apart. Suppressor requires that only one mover system is defined within each `LOCATION` category of the `PLAYER-STRUCTURE` template, which partly meets the above observations.

The example mover '1010 aew_airframe' has capabilities given by the named capability block 'aew_airframe_data' which begins as follows:

```
$ Capabilities for the aew_player's mover
CAPABILITY aew_airframe_data
  MAX-ACCELERATION    20.0 (M/SEC/SEC)
  MIN-TURN-RADIUS   1000.0 (M)
..<further optional capabilities of the mover>
END CAPABILITY
```

The two capabilities of maximum acceleration and minimum turn radius are both required for all mover systems. Both of these describe the limits on the rate of change of the player's velocity, `MAX-ACCELERATION` specifying the maximum rate of change of speed and `MIN-TURN-RADIUS` the maximum rate of change of direction.

Suppressor uses the turn radius to compute how tightly a player can execute a turn. Even though a mover's path may be pre-programmed as a sequence of points in space Suppressor does not normally[13] assume that the mover executes a series of straight paths from one to another, but rather that changes of direction follow smooth arcs. These arcs will respect the specified turn radius and are fully three-dimensional, i.e. they do not just represent the projected motion of the mover in the plane of the Earth's surface.

The setting of `MAX-ACCELERATION` controls how rapidly a mover may change speed in both reactive and pre-programmed motion. For this to take effect, however the `ACCELERATION-MODE` command found in the `TACTIC` block of the relevant player-type TDB `TACTIC` must be set as follows:

```
ACCELERATION-MODE  MAXIMUM
```

The default value of `UNIFORM` will effectively cause the maximum rate of change of speed to be ignored when movers are following pre-programmed paths. This kind of interdependence between different sections of the TDB is unfortunately quite common in Suppressor. It implies that considerable care must be taken in compiling Suppressor models so that an accurate representation of the model is achieved.

[13] The MODE: command of the SDB PATH item may be used to change this behaviour.

For player-types which can reactively manoeuvre three other capabilities can be set on the mover's speed, the rate of ascent and descent, and the minimum and maximum operating altitudes. The limits on the mover's are set with the MOVER-SPEED-LIMITS:

```
MOVER-SPEED-LIMITS
  MIN-SPD  100.0 (M/SEC)
  MAX-SPD  670.0 (M/SEC)
END MOVER-SPEED-LIMITS
```

Both the minimum and maximum speed limit must be given. The speed limits are required for players which can either avoid or follow terrain or avoid threats. For other systems these limits are optional.

The same movers that require MOVER-SPEED-LIMITS will also need to have their MOVER-CLIMB/DIVE-LIMITS set. These limit how rapidly an aircraft may change altitude when avoiding terrain or threats, or following terrain.

```
MOVER-CLIMB/DIVE-LIMITS
  MAX-DIVE-RATE    40.0 (M/SEC)
  MAX-CLIMB-RATE   80.0 (M/SEC)
END MOVER-CLIMB/DIVE-LIMITS
```

If these limits are omitted for a player which should have them then the player will not be able to change its altitude in performing its manoeuvres.

Any player which can reactively manoeuvre requires its maximum and minimum operational altitudes be set, these being given relative to mean sea level in the following form:

```
MOVER-ALTITUDE-LIMITS
  MIN-ALT  0.0 (M)
  MAX-ALT  20000.0 (M)
END MOVER-ALTITUDE-LIMITS
```

In the above examples values have been given using the SI units of measure, often Suppressor allows other units to be used; frequently standard imperial and nautical measures are available. The reference manual specifies the units that can be used with each command in full.

A mover system may consume fuel and if so a FUEL-USAGE table must be specified when it does, this table tabulating the rate of consumption of fuel as a function of altitude and speed:

```
FUEL-USAGE
  DIMENSION 1 ALT (M)   0.0   1000.0   2000.0   6000.0[14]
    DIMENSION 2 SPEED (M/SEC)   0.0   200.0   400.0   600.0
      BURN-RATE (KG/SEC)         0.112   0.117   0.125
    DIMENSION[15] 2 SPEED (M/SEC)   0.0   300.0   650.0
      BURN-RATE (KG/SEC)         0.114   0.124
    DIMENSION 2 SPEED (M/SEC)   0.0   225.0   450.0   775.0
      BURN-RATE (KG/SEC)         0.113   0.119   0.124
END FUEL-USAGE
```

The DIMENSION argument is frequently used to format data tables and some explanation of this is needed here. Its role is to assign values to hierarchical lists of user defined items. In fact the DIMENSION statement may be regarded as always introducing a list, although the list may only have one entry. Each of these lists is identified with an integer label, so that lists with the label 2 are used to refine the entries of the most recent list with label 1, and so on. Sometimes, as with the above example, the entries are actually bracketed intervals of a continuous real variable. In this case a list of *n* numbers would correspond to *n-1* entries. Only one list with label 1 ever occurs in each table, and for each entry in this list one list with label 2 may occur, and so on until the table is completely specified and the values of the input data given.

In the above table the lists go two levels deep, and three lists with label 2 provide speed intervals corresponding to the three altitude intervals specified in the first top level list. Each list of speed intervals must contain at least one interval, and for each interval a fuel burn rate is specified. So each burn rate corresponds to some pair of speed of altitude intervals. Similarly for a table with five levels of lists the final entry would correspond to a unique combination of five entries and so on. From the above table we can see, for example, that the mover would burn fuel at a rate of 0.119 kgs^{-1} when flying at a height of 4000 m above sea level and at a speed of 300 ms^{-1}.

Two more optional data items may be used to define mover systems. The first, NAV-ERROR-DATA allows the effect of navigational errors to be modelled. Movers using this data item will deviate from their intended paths in a manner which mimics the effects of navigation errors arising from misaligned magnetic compasses. If the user assumes that all the players in the scenario never make such errors then this item will not be required.

[14] There are four limiting altitudes defining three initial ranges, from 0 to 1 km, to 2 km, and to 6 km. Each of these ranges has a corresponding DIMENSION 2 statement giving the fuel usage rate whilst moving in these altitude ranges.

[15] This list entry corresponds to the second altitude range of 1 to 2 km. It has three speeds defining two speed ranges in its list, each with a corresponding burn rate. These are 0.114 kgs^{-1} from below 300 ms^{-1} and 0.124 kgs^{-1} above 300 ms^{-1}.

The size of the errors are based on the local magnetic dip angle which is set by the MAG-DIP-ANGLE entry in the SDB initialization entry. When this entry is absent or zero no navigational errors will be made. (Another example of interdependence between different portions of the user's input data.) An example table is shown below which each entry describing one of the various classes of navigational error. The heading error will be selected uniformly from the range specified by MIN-BASE-HDG to MAX-BASE-HDG and the speed will vary by a multiplicative factor chosen from the range specified by MIN-SPEED-ERROR and MAX-SPEED-ERROR. (So that if the required speed were 200 ms^{-1} and if the chosen factor were 1.10 the resulting speed would be 220 ms^{-1}.)

```
NAV-ERROR-DATA
  MIN-BASE-HDG     -2.0   (DEG)
  MAX-BASE-HDG      2.0   (DEG)
  VERT-AX-MEAN      3.0   (DEG)
  VERT-AX-DEV       0.5   (DEG)
  YAW-ALIGN-MEAN    4.0   (DEG)
  YAW-ALIGN-DEV     0.6   (DEG)
  MIN-SPEED-ERROR   0.9   (NO-UNITS)
  MAX-SPEED-ERROR   1.15  (NO-UNITS)
END NAV-ERROR-DATA
```

The errors in the mover's vertical axis and yaw angle are modelled using Gaussian distributions, the means being VERT-AX-MEAN and YAW-ALIGN-MEAN and the standard deviations VERT-AX-DEV and YAW-ALIGN-DEV.

The final optional entry of the mover's capability block is COMMIT-ALT, which is yet another example of interdependence between different portions of Suppressor. This allows the tactical behaviour of the mover's element to be modified according to its current dive or climb angle. It is possible, using the REL-TGT-ALT criterion of the player's tactics, see section 3.6.1, to alter these tactics according to the relative altitudes of the mover and its target. For example if the criterion appears as follows in the player's tactics:

```
WHEN REL-TGT-ALT < -600.0 (M)
```

then this means 'do something when less than 600 m higher than the target'. The purpose of COMMIT-ALT is to change the value of this height difference according to the current dive angle of the mover. (The name arises from this being the relative *altitude* at which the tactic *commits* the element to act.)

An example referring to the above test could be:

```
COMMIT-ALT
  DIMENSION 1 DIVE-SLOPE   (DEG)
       -90.0  -10.0  10.0  90.0
    DECIDE-ALT  (M)
       -800.0  -600.0  -400.0
END COMMIT-ALT
```

which would substitute a height difference of 400 m for 600 m when the dive angle was steeper than 10°, and 800 m for 600 m when the player was climbing at an angle greater than 10°. Otherwise the height difference of 600 m would be unchanged.

3.4.2 Weapons

Weapon systems provide a player with the ability to shoot at a target, i.e. to attack or lethally engage the target. We have already remarked that Suppressor may model weapons in two different ways. The first, and simplest, method uses weapons to fire ordnance (i.e. ammunition) at a target so that we might have:

```
WEAPON       2040 gun
  CAPABILITY      gun_data
  ORDNANCE        bullet   DISCRETE 100 (ROUNDS)
```

In this case nothing more is done than checking the speed and the trajectory of the bullets to see if they hit the target or not. Suppressor only uses information contained in the 'gun_data' capability block to predict the outcome of the event. The second way is to have the weapon launch new players, each complete with their own plans and tactics. For example the system definition:

```
WEAPON        2041 missile_launcher
  CAPABILITY       missile_launcher_data
  FUTURE-PLAYER    missile DISCRETE 6 (COPIES)
```

represents a weapon which may launch up to six missiles, each becoming a player within the scenario.

Several of the data items used to define a weapon's capabilities are appropriate to both kinds of ordnance. For example, consider the start of the capability definition for the weapon '2040 gun':

```
$ Example of a gun's physical capabilities
CAPABILITY  gun_data
  WPN-CHARACTERISTICS
    3D-FLYOUT  IMPLICIT-FLYOUT
    LAUNCH-ENVELOPE-P(K)
    UNCONTROLLED
    NO-SELF-DESTRUCT
  END WPN-CHARACTERISTICS
  NUM-SIMULTANEOUS-ROUND  1 (NO-UNITS)
  <further capabilities of the weapon>
END CAPABILITY
```

The first entry, `WPN-CHARACTERISTICS`, in the above example describes the characteristics of the current weapon; it is optional with the six items that can be set all having default values. However, it is often useful to give all the appropriate definitions for clarity. The two options `3D-FLYOUT` and `IMPLICIT-ENVELOPE` are in fact redundant, since these values may not be changed and so may be safely omitted. The remaining options are as follows, (with the default values listed first):

- `INTERCEPT-ENVELOPE-P(K)` and `LAUNCH-ENVELOPE-P(K)` which specify when Suppressor checks the values in the probability of kill, P_K, table. The choice is between Suppressor computing the chance of success at the time of intercept, i.e. when the ordnance hits the target, which is the default, or when the ordnance is launched. (The latter being the case in the above example.)
- `CONTROLLED` or `UNCONTROLLED` specifies whether the ordnance is guided by the player after it is launched. A controlled weapon must be guided by a linked tracking sensor. An uncontrolled weapon, such as our simple gun, has no control over the ordnance once it is fired.
- `ABORT-SALVO-WHEN-COASTING` and `CONTINUE-SALVO-WHEN-COASTING` describe whether or not a salvo will continue should the linked tracking sensor go in to a coasting mode. If the salvo does continue it will cease once the `MAX-COAST-TIME` defined in the `SNR-TIME-DELAYS` of the tracker's capability block is exceeded.
- `NO-SELF-DESTRUCT` and `SELF-DESTRUCTION` determine whether or not the weapon (and its associated location) are destroyed on impact. Weapons such as missile warheads will cause the missile they are carried in to self-destruct, bullets fired from guns will not.

Another optional characteristic of the weapon is specified with the NUM-SIMULTANEOUS-ROUND data item. This gives the number of targets the weapon can simultaneously engage. For the weapon '2040 gun' the value of this is just one:

```
NUM-SIMULTANEOUS-ROUND  1 (NO-UNITS)
```

which would also be the default value if the item were omitted.

When a weapon such as a gun is modelled by Suppressor some approximation must be made of the path of the ordnance. This will, first of all, allow the model to determine the likelihood of a hit and, should the ordnance impact the target, the position and time of the intercept point. The approximation is made via a look-up table, WPN-SPD-CAPABILITY, which tabulates the speed of the ordnance as a function of time and the relative altitude of the weapon with respect to the target. The simplest form of the table will be independent of altitude and, for the gun described above, is:

```
WPN-SPD-CAPABILITY
  DIMENSION 1  TIME (SEC) 0.0  7.0
    AVG-SPD (M/SEC)        2000.0
END WPN-SPD-CAPABILITY
```

If the weapon's performance depends on altitude one of two further forms may be used, for example:

```
WPN-SPD-CAPABILITY
  DIMENSION 1  REL-TGT-ALT  (M)
    0.0  20.0  80.0  50.0E3
    DIMENSION 2  TIME (SEC) 0.0  2.0  1000.0
      AVG-SPD (M/SEC)         10.0  20.0
    DIMENSION 2  TIME (SEC) 0.0  2.0  4.0  1000.0
      AVG-SPD (M/SEC)         10.0  30.0  40.0
    DIMENSION 2  TIME (SEC) 0.0  2.0  4.0  6.0  1000.0
      AVG-SPD (M/SEC)         10.0  30.0 50.0  60.0
END WPN-SPD-CAPABILITY
```

which represents a bomb dropped from rest under gravity; it accelerates to 60 ms^{-1} and then drops at this terminal velocity. The extra dependency on the relative altitude of the player and its target is needed to prevent spurious acceleration when the target is some distance away in a horizontal direction. Despite this the path of the bomb is not accurately modelled by this entry since Suppressor will always assume that it can always directly head towards the target even when dropped at a considerable range from the target. This is in fact not so important as it may seem, since Suppressor does not actually compute the path of the bomb but only uses the WPN-SPD-CAPABILITY table to infer if an intercept is even possible before the weapon reaches its maximum range. The weapon's maximum ranges are implied by the table's entries, with a different maximum range corresponding to each row of the WPN-SPD-CAPABILITY table. The maximum range for each altitude interval $R_{\max}$ is obtained as $R_{\max} = \sum_i s_i \delta t_i$ where s_i is the speed entry of column i and δt_i is the duration for which this average speed is maintained. Should the target be beyond this range when the weapon is fired the intercept will not occur and the ordnance is lost. A message, `CANNOT-INTERCEPT`, will be recorded in the simulation log file, (the 'Time History' file printed out by the model execution MOD phase, see the reference manual for details).

Another alternative form of the `WPN-SPD-CAPABILITY` data item uses the absolute height of the target above ground level, `TGT-ALT-AGL`, instead of the relative altitude of the player and target, `REL-TGT-ALT`.

It may be more accurate to include the initial speed of the ordnance in its flight. In general this is imparted to it by the motion of the weapon itself, which in turn is given by the speed of its 'platform'. To avoid having to compute this value individually for all the weapons it may be included in the computation of the ordnance's motion by setting the optional data item `PLATFORM-VEL-ATTEN` to one. If this entry is not set then Suppressor will not include the initial motion, (so that the bomb described above will always drop directly towards the target.) If the value is set as follows:

```
PLATFORM-VEL-ATTEN    1.0   (NO-UNITS)
```

then at least the imparted motion of the weapon will be included correctly.

In addition to the time taken for the ordnance to physically intercept the target after it is fired additional physical delays occur before the ordnance leaves the weapon. These represent the time taken for the weapon to discharge the ordnance it is firing. For the '2040 gun' weapon values are set using the `WPN-TIME-DELAYS` item as follows:

```
WPN-TIME-DELAYS
  SHOOT-TIME-DELAY    0.45 (SEC)
  SALVO-FIRING-DELAY  0.25 (SEC)
END WPN-TIME-DELAYS
```

Two types of delay may be set: the first SHOOT-TIME-DELAY indicates the interval between deciding to use a weapon and the launch of the first round. If salvoes are being fired which launch more than one round subsequent intervals between the firing of rounds are given by the second delay, SALVO-FIRING-DELAY. Both of these entries have default values of zero. Another data item, the WPN-TIME-DELAY-TABLE, allows the SHOOT-TIME-DELAY to be changed when the target is protected by an electronic counter-measures (ECM) system such as a jammer. This table has the following format:

```
WPN-TIME-DELAY-TABLE
  DIMENSION 1  JAMMER-TYPE DEFAULT jammer
    DIMENSION 2  REL-TGT-ALT  (KM)  -10.0   10.0
      SHOOT-TIME-DELAY   1.50 (SEC)
    DIMENSION 2  REL-TGT-ALT  (KM)  -10.0   10.0
      SHOOT-TIME-DELAY   7.50 (SEC)
END WPN-TIME-DELAYS
```

A list syntax is used with the first list giving the names of jammers, both implicit and explicit, which can influence the firing times. The entry DEFAULT covering all disruptors not otherwise named must always be present. The second list defines intervals describing the weapon's altitude relative to the target, the above example using just one interval for the whole range of possible heights. Finally the SHOOT-TIME-DELAY which is used to over-ride the same entry in the WPN-TIME-DELAY item in the presence of effective jamming from the named disruptor is given.

Once ordnance has been fired, and an intercept has been computed, Suppressor must model the damage, if any, incurred by the target. This task is facilitated by the weapon's probability of kill, P_K, table. This, as might be imagined, is a complex data entry in Suppressor with many possible options arranged in tabular format. For the simple gun described above the P_K entry could be quite simple with the table being introduced by the keyword WPN-PK and being in the form:

```
WPN-PK
  DIMENSION 1 ELEMENT-TYPES  DEFAULT
    PK (NO-UNITS) 0.90
END WPN-PK
```

In other words, against any target elements the gun will achieve a P_K of 90%.

More generally, the WPN-PK table may have several, in fact up to eighteen, different lists labelled using the DIMENSION statement. All of which, except the ELEMENT-TYPES list, being optional. The format of the WPN-PK is in fact unique. Although there are many other tabular entries in the Suppressor data files all either have a fixed format for the ordering and numbering of the DIMENSION lists, like the FUEL-USAGE block; or only allow a small number of variants, like the WPN-SPD-CAPABILITY table. In the case of the WPN-PK table so many options occur that apart from the ELEMENT-TYPES option, which must be the first, the use and ordering of the options is determined by the user. Two further restrictions do occur though, firstly that no option is used more than once and secondly that once an ordering is set it must be maintained throughout the remainder of the table.

As an example of this consider the P_K table appropriate for some weapon '9043 hand_grenade'. The hand grenade may be very effective against personnel but useless when thrown against tanks, so this hypothetical weapon could have a WPN-PK table of the form:

```
WPN-PK
  DIMENSION 1 ELEMENT-TYPES  infantryman tank DEFAULT
    DIMENSION 2 RNG 0.0 20.0 30.0 40.0 (M)
      PK (NO-UNITS)  0.90  0.50  0.30
    DIMENSION 2 RNG 0.0 1000.0 (M)
      PK (NO_UNITS) 0.0
    DIMENSION 2 RNG 0.0 1.000.0 (M)
     PK (NO-UNITS)  0.0
END WPN-PK
```

Because (against troops) the weapon's effectiveness depends strongly on range the P_K table reflects this, but now this entry must also be included for both tanks and all other targets, even though the weapon is ineffectual at all ranges. So if some option is required for just one class of target it must be included, however pointless, for all.

This problem can be minimized with the help of a WPN-PK-DEGRADE table which introduces a degradation factor in exactly the same manner as the WPN-PK table introduces the initial P_K value. So instead of the above WPN-PK
table we could instead have

```
WPN-PK
  DIMENSION 1 ELEMENT-TYPES  infantryman DEFAULT
    DIMENSION 2 RNG 0.0 20.0 30.0 40.0 (M)
      PK (NO-UNITS)  0.90  0.50  0.30
    DIMENSION 2 RNG 0.0 1.000.0 (M)
     PK (NO-UNITS)  0.0
END WPN-PK
```

and a further table of weapon P_K degradations factors for the hand grenade's use against tanks:

```
WPN-PK-DEGRADE
  DIMENSION 1 ELEMENT-TYPES  tank  DEFAULT
    DEGRADE-FACTOR (NO-UNITS) 0.0    1.0
END WPN-PK-DEGRADE
```

The final P_K is the product of the initial P_K from the WPN-PK table and the degradation factor from the WPN-PK-DEGRADE table. This method does not save much for the above, simple example, but usually offers an easier way of modifying a P_K for a single target than introducing a new `ELEMENT-TYPES` entry in the original table, when this element depends on different variables than the existing elements.

The available options for both the `WPN-PK` and the `WPN-PK-DEGRADE` tables and their meanings are listed below:

- `ELEMENT-TYPES` is the required first entry in the table. It lists the names of the target elements affected by the weapon and the compulsory entry `DEFAULT`.
- `OFFSET` specifies the distance between the target and the weapon in the horizontal plane. A positive offset implies the target is to the weapon's left; a negative offset that it is to the weapon's right.
- `ABS-OFFSET` may be used instead of `OFFSET` when there is a bilateral symmetry in the weapon's performance. In this case all horizontal offsets are represented by positive numbers.
- `RNG` specifies the range of the target from the weapon. Its meaning changes depending on the presence of either of the `OFFSET` entries. If horizontal offsets are not specified then `RNG` means the two dimensional range in the horizontal plane of the Earth's surface. When either `ABS-OFFSET` or `OFFSET` are specified `RNG` means the up/down range of the target. Here up/down refers not to vertical separations but to the distance of the target from its projected point of closest approach to the weapon. When it is approaching this point the range of the target will be positive, when receding the range will be negative.
- `ALT` may be used to specify the vertical separation of the target and the weapon. A positive value means that the target is above the weapon, a negative value that it is below the weapon.
- `TGT-ALT-AGL` specifies the absolute altitude of the target above ground level. When terrain is not modelled it means the height above mean sea level.
- `REL-ST-DIVE-ALT` is the relative altitude of the attacker to the target when it commenced its final dive towards the target.
- `TGT-SPD` is the relative speed of the target towards the weapon. Positive values mean that the target is moving towards the weapon, negative values that it is going away.
- `VERTICAL-SPD` is the speed of the target in the vertical direction. A positive value implies that the target is climbing, a negative one that it is descending.
- `REL-TGT-EL-ANG` is the relative elevation angle from the weapon to the target. Positive values mean the target is above the weapon, negative values that the target is below.
- `HDG-CROSS-ANGLE` is the angle between the velocity vectors of the target and the weapon.
- `REL-TGT-HDG` is the angle formed by the velocity vector of the target and the range vector from the target to the weapon.
- `WPN-VEL-EL-ANG` is the vertical angle of the ordnance at the time of intercept.
- `ELAPSED-COAST-TIME` is the elapsed time since the weapon's tracking systems lost their lock on the target and started to coast. As this time increases the value of the P_K will diminish for the weapon.

- `COLLATERAL-ELEMENTS` gives P_K values for elements other than those explicitly targeted but still lying at the same location.
- `JMR-TYPES` identifies the influence the presence of an effective disruptor of the named type will have upon P_K. A disruptor is only effective if it is both operational and is currently disrupting the weapon's tracking radar. Disruptors need not be jammers to be effective here, even simple disruptors modelling chaff or flares can be used to degrade the weapons P_K value.
- `ORDNANCE-TYPES` lists the P_K for the various kinds of ordnance the weapon may employ.
- `TRACKER-TYPES` gives the names of the tracking sensor receiver used in the `WITH-TRACKER` clause in the `LETHAL-ENGAGE-START` procedure of the `RESOURCE-ALLOCATION` tactical block. The keyword `NONE` indicates the absence of the tracker in the engagement.

A more complex example of a P_K table is provided below:

```
WPN-PK
  DIMENSION 1 ELEMENT-TYPES  DEFAULT
    DIMENSION 2 JMR-TYPES  jukebox   DEFAULT
      DIMENSION 3 ALT (M) 0.0  20.0  5000.0  10000.0
        DIMENSION 4 RNG (KM) -60.0  60.0
          PK (NO-UNITS)  0.0
        DIMENSION 4 RNG (KM) -60.0  -40.0  40.0  60.0
          PK (NO-UNITS)  0.0   0.31   0.0
        DIMENSION 4 RNG (KM) -60.0   60.0
          PK (NO-UNITS)  0.0
      DIMENSION 3 ALT (M) 0.0  20.0  5000.0  10000.0
        DIMENSION 4 RNG (KM) -60.0  60.0
          PK (NO-UNITS)  0.05
        DIMENSION 4 RNG (KM) -60.0  -40.0  40.0  60.0
          PK (NO-UNITS)  0.02   0.72   0.02
        DIMENSION 4 RNG (KM) -60.0   60.0
          PK (NO-UNITS)  0.05
END WPN-PK
```

where the probability of kill is modified by the presence of a jammer, the relative altitude of the target and its horizontal range.

As weapons consume their ammunition it is possible with Suppressor to model the reloading of the weapon. This is done with the RELOAD-CHARACTERISTICS table. If this optional entry is included the weapon will be reloaded once its remaining ammunition drops below some threshold value should further stocks of the ammunition remain. After some time delay, (once the weapon's ammunition is replenished), the weapon is returned to normal operation. The element's tactics may be modified to take into account the possibility that a weapon is reloading by checking the RELOAD-STATUS criterion within the RESOURCE-ALLOCATION tactics block. The '2040 gun' weapon may be reloaded with the help of:

```
RELOAD-CHARACTERISTICS
  DIMENSION 1 AMMO-TYPE   bullet
  EXTRAS (ROUNDS) TIME (SEC) THRESHOLD (ROUNDS) RELOAD (ROUNDS)
    500              40          10                    80
END RELOAD-CHARACTERISTICS
```

This will replenish the original stock of 100 rounds with 80 fresh bullets from a store of 500 spare rounds once the number remaining has fallen to ten rounds or less. The reloading process will leave the gun unavailable for 40 seconds however.

As yet we have not discussed those weapon systems which spin off FUTURE-PLAYERS as ordnance. All the above data items are available for weapon which launch FUTURE-PLAYERS. In practice the weapon itself requires less data items, since the WPN-PK table and the WPN-SPD-CAPABILITY tables are not required. Now the modelling of these items will be taken over by the new player. One extra command that is required is that which equates the FUTURE-PLAYER resource to the player-type that is created, (these names are not the same). This is accomplished with a RESOURCE-DISAGGREGATION table, which for the example '2041 missile_launcher' given above is:

```
RESOURCE-DISAGGREGATION
  DIMENSION 1 RESOURCE-TYPE missile_fly
    CREATED-PLAYER missile_player
END RESOURCE-DISAGGREGATION
```

This means that the FUTURE-PLAYER 'missile_fly' once used by the missile launcher cause the creation of a new player whose type is that of a 'missile_player'. This player-type must be defined as a PLAYER-STRUCTURE template within the TDB. Furthermore one example of such a player must be named and located in the appropriate place in the missile launcher's command chain ready for use. (This is done using the SDB PLAYER: command.)

The missile player itself will have its own weapon, for example:

```
WEAPON 1040 missile_warhead
  CAPABILITY   missile_warhead_data
  ORDNANCE     tnt DISCRETE  1 (ROUNDS)
```

a missile warhead armed with high explosive. This weapon will have appropriate P_K and weapon speed tables defined in exactly the manner described above.

3.4.3 Thinkers

In discussing the nature of Suppressor's event processing in section 2 it was made clear that thinkers play a special role. The thinkers represent the cognitive agents, usually human beings, who not only perceive but direct and control the course of each simulation. The process of thinking divides into three distinct stages. The first is *noticing* where an external stimulus is first noted by a thinker and placed in short-term memory. The second stage is *digestion* or *fusion* where this perception updates the thinker's current perceptions, these being stored in medium term memory. The final stage is *reaction* where the thinker decides, what, if any, action to take given its current knowledge of the situation. The role of each thinker's capability block is to determine the following:

- how long a thinker will retain its short and medium term perceptions of the scenario;
- which events it is capable of thinking about;
- how long the thinker will take to think about each event;
- how many different events it can consider simultaneously;
- the nature of any FUTURE-PLAYERs it may launch as subordinates.

The most important entries in determining the above thinker capabilities are the TIME-BEFORE-DROP and the TIME-TO-THINK data items. Take for example the AEW craft's thinkers:

```
THINKER         1000 aew_commander
  CAPABILITY         aew_commander_think_data
  FUTURE-PLAYER      interceptor DISCRETE 3 (COPIES)
THINKER         1001 aew_operator
  CAPABILITY         aew_operator_think_data
```

Since there are two thinkers their responsibilities and capabilities must be both discriminated and determined by their capability entries.

Firstly the thinker '1001 aew_operator' is linked to the AEW craft's radar so this thinker should be given special responsibilities in this regard. The TIME-TO-THINK entry is used to set the time each thinker will take to think about some particular circumstance. Three entries are used to recognise different kinds of events; these enable a thinker to carry out the first kind of thinking modelled by Suppressor, i.e. noticing. The AEW operator must be able to notice sensor detections placed in the player's iconic memory, so one entry in the TIME-TO-THINK item must be RECOG-SNR-EVENT. This allows a thinker to recognise a sensing chance, hence the '1001 aew_operator' must have set:

```
CAPABILITY  aew_operator_think_data
  TIME-TO-THINK
    RECOG-SNR-EVENT      0.13 (SEC)
    <Omitted thinking times>
  END TIME-TO-THINK
  <Omitted thinker capabilities>
END CAPABILITY
```

So we have set that the operator will recognise a sensing chance, i.e. place the detection in short term memory. Once the event has been noticed its implications must still be digested. That is to say the new information must be used to update the player's current perceptions in medium term memory. This could be done by either the commander or the operator and is enabled with the ASSIMILATE-INTELL entry. So the operator's TIME-TO-THINK items now include:

```
TIME-TO-THINK
  RECOG-SNR-EVENT      0.13 (SEC)
  ASSIMILATE-INTELL    0.21 (SEC)
  <Omitted thinking times>
END TIME-TO-THINK
```

If the operator thinker is solely concerned with the radar sensor this could complete its TIME-TO-THINK entries, but the ability to periodically review and update all its perceptions independently of sensing stimuli, which at least one thinker must have, could also be included. Furthermore, the operator may also represent the radio operator, giving the thinker double duty, so that in this case the ability to recognise incoming messages would also be required. These two abilities are conferred by the REVIEW-INFORMATION and the RECOG-MSG items, so that we now have:

```
TIME-TO-THINK
  RECOG-MSG             0.12 (SEC)
  RECOG-SNR-EVENT       0.13 (SEC)
  ASSIMILATE-INTELL     0.21 (SEC)
  REVIEW-INFORMATION    0.20 (SEC)
END TIME-TO-THINK
```

At this stage the operator can handle sensor detections and messages and process these to incorporate them in the player's medium term memory. Overall, however, the AEW craft should also have command responsibilities, to assign subordinates to potential targets and also to launch subordinates which, for example, could cause interceptors on alert to scramble from an airbase. These responsibilities should be carried by the '1000 aew_commander' thinker, with the following capability block:

```
CAPABILITY  aew_commander_think_data
  TIME-TO-THINK
    RECOG-PHYS-EVENT        0.13  (SEC)
    ASSIMILATE-INTELL       0.18  (SEC)
    REVIEW-INFORMATION      0.30  (SEC)
    CONSIDER-LAUNCH         3.20  (SEC)
    CONSIDER-ASG/CANCEL     2.00  (SEC)
    CONSIDER-MOVE           1.50  (SEC)
    EVAL-ASSIGN-THREAT      3.75  (SEC)
    EVAL-GUNS-FREE/TIGHT    2.50  (SEC)
    EVAL-EMCON-CHANGE       2.00  (SEC)
  END TIME-TO-THINK
  <Omitted thinker capabilities>
END CAPABILITY
```

The commander has a much more extensive list of responsibilities than the operator in this example. Firstly, the commander must be responsible for noticing all physical stimuli other than incoming messages and sensing chances. In Suppressor this means noticing that an attack is being made on the player and is set with the RECOG-PHYS-EVENT item. This information must be digested and reviewed which lead to the corresponding settings of the thinker's capabilities. In addition to noticing and digesting, which the operator can also do, the commander must also react, i.e. make decisions based on its perceptions.

The first of these decisions are associated with assigning subordinates to engage targets, known as LETHAL-ASSIGNMENT. The commander first evaluates each perceived target to see if it is a candidate for a lethal assignment which consists of asking the question: 'is this target one I should consider assigning a subordinate to attack?'. The entry EVAL-ASSIGN-THREAT confers this responsibility. If the answer to this question is yes then the commander can consider making a lethal assignment to a subordinate, or launching a subordinate to intercept this threat. The commander may decide not to take either action if the threat posed is too slight or if there are no resources available for use. These processes will take the time specified by the CONSIDER-ASG/CANCEL and CONSIDER-LAUNCH items.

If a subordinate has been assigned to attack a target the commander may wish to give the subordinate the authority to engage the target on its own, or it may seek to retain the control. Either way the decision to set the subordinates 'lethal mode of control' will take the time specified by the EVAL-GUNS-FREE/TIGHT entry.

Finally, two further items must be specified if the AEW craft is to be able to manoeuvre reactively and to reactively turn on and off its sensor receiver, these are CONSIDER-MOVE and EVAL-EMCON-CHANGE respectively.

In addition to the responsibilities of thinking about certain stimuli and taking cetain actions the two thinkers need to know how long to keep old perceptions before discarding them. In short-term memory (after a stimulus has been noticed but before it has been digested) a perception will be kept until either a new perception of the same nature displaces it or the perception is older than a specified age. This time limit is set by the TIME-BEFORE-DROP entry and is required for all thinkers which assimilate intelligence. It also helps control the retention of perceptions in medium term memory. Once the perception has been incorporated into the player's 'world view' of the scenario it is placed in medium term memory and cannot be retired quite so easily as it could from short term memory. A perception of a target will though be dropped from medium-term memory if there are no assignments made on that target and if the perception is older than the time specified by TIME-BEFORE-DROP.

By default, each thinker can only process one event at once, but we have seen that each thinker may represent many crew members. Hence, it may be appropriate to allow the operator to handle two things at once, since it represents both the radar and radio operators. This can be accomplished with the MAX-CONCURRENT-EVENTS data item. So in summary we may have the following capability blocks for the two thinkers:

```
CAPABILITY  aew_operator_think_data
  TIME-TO-THINK
    RECOG-SNR-EVENT       0.13  (SEC)
    <Omitted thinking times>
  END TIME-TO-THINK
  TIME-BEFORE-DROP       60.0  (SEC)
  MAX-CONCURRENT-EVENTS 2     (NO_UNITS)
END CAPABILITY
```

with the commander's capabilities being given by:

```
CAPABILITY  aew_commander_think_data
  TIME-TO-THINK
    RECOG-PHYS-EVENT      0.13  (SEC)
    <Omitted thinking times>
  END TIME-TO-THINK
  TIME-BEFORE-DROP       60.0  (SEC)
  RESOURCE-DISAGGREGATION
    DIMENSION 1 RESOURCE-TYPE interceptor
      CREATED-PLAYER fighter_player
  END RESOURCE-DISAGGREGATION
END CAPABILITY
```

Two things to note are that since the player's memory is shared between the thinkers they both should have the same TIME-BEFORE-DROP and that the commander may launch players of the type 'fighter_player' as subordinates.

The fighter player type will have the job of engaging its targets, so it will need at least one thinker which can process mental activities associated with lethal engagements. For this example suppose that the fighter has thinkers defined as follows:

```
THINKER     1000 fighter_pilot
  CAPABILITY     fighter_pilot_think_data
```

with the selected capabilities given below:

```
CAPABILITY  fighter_pilot_think_data
  TIME-TO-THINK
    RECOG-PHYS-EVENT      0.15  (SEC)
    <Omitted thinking times>
    EVAL-ENGAGE-THREAT    0.32  (SEC)
    EVAL-LETHAL-ENGAGE    0.15  (SEC)
    EVAL-FIRING           0.12  (SEC)
  END TIME-TO-THINK
  TIME-BEFORE-DROP       100.0  (SEC)
END CAPABILITY
```

Several thinking times associated with noticing, digestion and reacting have been omitted, but three new processing times concerned with LETHAL-ENGAGE tactics are present. The first of these reactive thinking capabilities, EVAL-ENGAGE-THREAT, determines if a target is actually eligible to be engaged. The second, EVAL-LETHAL-ENGAGE, determines how long the thinker will take to consider beginning or ending a lethal engagement against a single target. The third thinking time is how long the pilot takes to think about opening or ceasing fire on a target.

In addition to the fourteen thinking times discussed so far two more times exist which relate to NONLETHAL-ENGAGEMENT tactics i.e. disrupting or jamming. A player which has the generic function of disrupting will require at least one thinker which can consider adding a target to a list targets which might be jammed, EVAL-JMR-QUEUE and which can also think about starting or stopping the jamming of such targets, EVAL-JMR-SPOTS.

A full list of thinking times and the responsibilities with which they are associated is given in the following tables, the first table deals with the 'noticing' mental processes.

Noticing: noticing places information in a player's short-term memory. It is the first stage in the thinking process and involves the thinker's recognition of external stimuli.

Thinking Time	Activity
RECOG-MSG	Notice message
RECOG-PHYS-EVENT	Notice physical stimulus (e.g sense attack)
RECOG-SNR-EVENT	Notice sensing chance

Digesting: digesting merges new perceptions into medium-term memory and keeps the player's perceptions current. It is the second stage of the thinking process. The user has relatively little control over this phase.

Thinking Time	Activity
ASSIMILATE-INTELL	Digest noticed events, (i.e. stimuli)
REVIEW-INFORMATION	Update perceptions periodically

Reacting: reacting is the final stage of the thinking process and in this phase the thinker will decide what to do next. Here the user has almost complete control of the process since just what will be done is determined by the player-type's tactics. One or more of seven possible tactical responses can be selected if the thinker has the appropriate thinking times defined for them. The following tables indicate which thinking times correspond to which user defined tactics from the `TACTIC` block of the TDB:

- Reactive Manoeuvres: used by players which can move off a pre-programmed path.

Thinking Time	Associate Tactics
CONSIDER-MOVE	MOVE-PLANS

- Lethal-Assignments: used by a commander to assign subordinates to engage a target.

Thinking Time	Associated Tactics
EVAL-ASSIGN-THREAT	LETHAL-ASSIGNMENT-QUEUE-ADD LETHAL-ASSIGNMENT-QUEUE-DROP
CONSIDER-ASG/CANCEL	LETHAL-ASSIGNMENT-START LETHAL-ASSIGNMENT-STOP

- Lethal Engagements: used by players which can engage a target.

Thinking Time	Associated Tactics
EVAL-ENGAGE-THREAT	LETHAL-ENGAGE-QUEUE-ADD LETHAL-ENGAGE-QUEUE-DROP
EVAL-LETHAL-ENGAGE	LETHAL-ENGAGE-START LETHAL-ENGAGE-STOP
EVAL-FIRING	LETHAL-ENGAGE-FIRING-START LETHAL-ENGAGE-FIRING-STOP

- Non-lethal Engagements: used by players which can disrupt a target's functional systems.

Thinking Time	Associated Tactics
EVAL-JMR-QUEUE	JAMMER-QUEUE-ADD JAMMER-QUEUE-DROP
EVAL-JMR-SPOTS	JAMMER-SPOT-ADD JAMMER-SPOT-DROP

- Emission Control: used by a player to control its radar sensor emissions:

Thinking Time	Associated Tactics
EVAL-EMCON-CHANGE	EMCON/TURN-ON EMCON/TURN-OFF

- Lethal Mode of Control: used by a commander to change the authority of a subordinate to engage a target.

Thinking Time	Associated Tactics
EVAL-GUNS-FREE/TIGHT	GUNS-FREE GUNS-TIGHT

- Launching Movement: used by a commander to launch a subordinate into motion.

Thinking Time	Associated Tactics
CONSIDER-LAUNCH	LAUNCH-START

3.4.4 Sensors

Sensing systems allow a player to directly perceive other players in the scenario. Suppressor models four classes of sensors: optical, infra-red, radar warning receivers and radars. Of these, all but radar systems are passive detection systems, requiring only `SENSOR-RECEIVERS`. Radars, however, are active and so in addition to a receiver they require one or more `SENSOR-TRANSMITTERS` . Our discussion of sensor systems will begin with sensor transmitters, and then generic sensor receivers. Finally details specific to individual classes of sensor receivers will be described.

3.4.4.1 Sensor Transmitters

Sensor transmitters can only operate in the radio frequency (RF) bands and are designed to operate in conjunction with radar receivers. A radar system consisting of a receiver, 'aew_radar_rx', and a transmitter 'aew_radar_tx' that are designed to work together must be explicitly linked in the player structure definition as follows:

```
SNR-RCVR      1030 aew_radar_rx
  CAPABILITY       aew_radar_rx_data
SNR-XMTR      1031 aew_radar_tx
  CAPABILITY       aew_radar_tx_data
LINKAGES    1030 WITH 1031
```

Other players may be able to listen in to sensor transmitters using their own warning receivers, which are essentially broadband receivers operating without a transmitter. Warning receivers form, from the viewpoint of Suppressor, a distinct class of sensor receivers.

Suppressor has coded into it some basic physical laws describing the propagation of electromagnetic radiation. This means that it can compute the power arriving at a receiver that was from radiated from a transmitter with given properties and then reflected by some surface at a distant point. For the transmitter Suppressor will take into account such things as power output, frequency, antenna gains, internal losses and atmospheric attenuation, Doppler effects, the consequences of pulsed transmissions. Suppressor will also check the line-of-sight (so that a transmitter or receiver may be shielded by a hill or over the horizon and so rendered ineffective even if within range) and the properties of the surface that is reflecting the radiation. However it is, in general, up to the user to ensure that all these values are specified consistently, since in most cases Suppressor will not check that the specified properties are consistent or even sensible.

The description of a transmitter in Suppressor can be exemplified with the system '1031 aew_radar_tx' defined below with the following physical capabilities:

```
CAPABILITY  aew_radar_tx_data
  PEAK-POWER-OUTPUT            5.0E6  (WATTS)
  XMIT-FREQ                    3.0E9  (HZ)
  INTERNAL-LOSS              -10.0    (DB)
  PEAK-GAIN                   30.0    (DB)
  ANTENNA-PATTERN
    DIMENSION 1 AZ    (DEG)    0.0  180.0
      DIMENSION 2 EL  (DEG)  -90.0   90.0
        GAIN          (DB)        30.0
  END ANTENNA-PATTERN
  PULSE-REPETITION-FREQ      250.0    (HZ)
  DUTY-CYCLE                   0.2    (NO-UNITS)
END CAPABILITY
```

The above transmitter is broadcasting a pulsed signal with a peak power of 5 MW at a frequency of 3 GHz. Each pulse cycle, i.e. the recurring cycle of transmission of a short pulse followed by a phase where the returned echo is listened for, recurs with a frequency of 250 Hz, implying that the complete pulse cycle takes 4 ms. The transmitter is on for one fifth of this cycle with a pulse duration of 0.8 ms. Thus, the average power of the transmitter is 1.0 MW. (Note that, most radars are pulsed since a single antenna functions both as a transmitter and a receiver making it necessary that the transmitter be switched off while the receiver is listening for radar echoes.) The transmitter also has internal losses of 10 dB, which means 10% of the generated power is actually emitted as radiant energy. The commands which describe these properties are described below, along with other options that are available for the capabilities of a `SENSOR-TRANSMITTER`.

The frequency at which the transmitter operates is defined by the `XMIT-FREQ` keyword. (This frequency may be over-ridden by the `FREQ:` entry in the SDB). Suppressor assumes that the transmission is coherent, i.e that its bandwidth is narrow. It has the capability to model pulsed Doppler radars, which if required is accomplished by setting the `PULSE-REPETITION-FREQ` and the `DUTY-CYCLE` for the transmitter, and also a moving target indicator (MTI) entry may be defined in the associated radar receiver's capability block. If MTI factors are not modelled then the pulse repetition frequency (PRF) is not needed by Suppressor. The `DUTY-CYCLE`, which specifies what fraction of a pulse cycle is spent transmitting energy, should always be set away from its default value of one for all pulsed radars, in order to accurately model the range of the system.

An important aspect of the definition of any radar transmitter (or receiver) is the antenna pattern. This determines the performance of the antenna as a function of angle. Suppressor provides several ways of defining this quantity. Realistic antennae do not perform equally well in all directions, although the example above, using the ANTENNA-PATTERN table, does. Its format is:

```
ANTENNA-PATTERN
  DIMENSION 1 AZ   (DEG)    0.0  180.0
    DIMENSION 2 EL (DEG)  -90.0   90.0
      GAIN          (DB)      30.0
END ANTENNA-PATTERN
```

Which means that all azimuths between -180° and 180°, and at all elevations between -90° and 90° the antenna has a gain of 30 dB. (Suppressor assumes that the antenna pattern has azimuthal symmetry unless otherwise given so that the range from -180° to 0° need not be explicitly specified.) This antenna could not exist in reality, since the gain of an antenna is the ratio of the power per unit solid angle received, or transmitted, in some direction to the power per unit solid angle that would be received if the antenna performed entirely uniformly in all directions. Therefore no antenna can give a real gain in performance in all possible directions. More realistic patterns may of course be encoded using this command or one of three standard antenna pattern generating functions may be used. The reasons why often a quite simple ANTENNA-PATTERN is all that is required for sensors will be discussed in the radar sensor receiver section below.

As remarked three standard generating functions for the antenna pattern can be used. The first is the LAMBDA-PATTERN which may be used for systems with circular antennae and uniform illumination. This generates a standard pattern of the form $(\sin(x)/x)^2$. The syntax is:

```
LAMBDA-PATTERN
  PEAK-GAIN        20.0   (DB)
  MIN-GAIN        -10.0   (DB)
  BEAMWIDTH         0.05  (RAD)
END LAMBDA-PATTERN
```

where the peak gain, PEAK-GAIN, the minimum gain, MIN-GAIN, and the beamwidth, BEAMWIDTH, of the antenna must be specified. It should be useful to recall here that the peak gain of an antenna G is given by $G = 4\pi A_e / \lambda^2$ where A_e is the effective area of the antenna and λ is the wavelength of the radiation. The beamwidth is the angular separation between the half power points of the pattern. For these antennae the first sidelobe will be 17.6 dB weaker than the main lobe. The gains well away from the mainbeam will be taken to be the minimum gain for all the generated antenna patterns.

Radars with a rectangular or elliptical aperture and a $(\sin(x)/x)^2$ pattern may be modelled with the SINE-PATTERN function. Here the first sidelobes are 23.5 dB weaker than the main lobe and the syntax of the command is of the form:

```
SINE-PATTERN
  PEAK-GAIN              20.00   (DB)
  MIN-GAIN              -10.00   (DB)
  AZ-BEAMWIDTH            0.05 (RAD)
  EL-BEAMWIDTH            0.08 (RAD)
  EL-BORESIGHT-ANGLE 0.00 (RAD)
END LAMBDA-PATTERN
```

Again the peak and minimum gains must be specified but in this case the beamwidths are no longer symmetric, and so the azimuthal and elevation beamwidths, AZ-BEAMWIDTH and EL-BEAMWIDTH,must be specified separately. Furthermore, the centre of the main lobe may be tilted up or down from its reference position with the EL-BORESIGHT-ANGLE entry.

Finally, a predefined pattern for rectangular or elliptical antennae with a cosecant squared illumination is available. Here the pattern is uniform in the azimuthal direction but diminishes according to a $\csc^2(x)$ law in elevation. Cosecant squared antennae are designed to maintain a constant return from a target which is approaching the radar at a constant altitude. This pattern is set using the COSECANT-PATTERN data item as follows:

```
COSECANT-PATTERN
  PEAK-GAIN                  20.00   (DB)
  MIN-GAIN                  -10.00   (DB)
  AZ-BEAMWIDTH                0.2    (RAD)
  MIN-EL-FOR-PEAK-GAIN        0.1    (RAD)
  PEAK/CSC2-BOUNDARY-EL 0.15   (RAD)
  MAX-EL-FOR-CSC2             0.3    (RAD)
END LAMBDA-PATTERN
```

Here, as with the other generating functions, PEAK-GAIN is the maximum gain of the antenna pattern and MIN-GAIN is its minimum gain. The antenna's performance is only described by the elevation parameters within a sector in the azimuthal direction with beamwidth given by AZ-BEAMWIDTH. Beyond this the gain is just the minimum gain everywhere. When the target as an azimuth which is less than this the gain is described by three elevation angles which will now be described. MIN-EL-FOR-PEAK-GAIN is the minimum elevation at which the gain of the antenna is the peak gain; below this elevation it is the minimum gain. This angle must be greater than or equal to zero degrees. PEAK/CSC2-BOUNDARY-EL is the elevation angle which divides the part of the antenna pattern which has the peak gain from the cosecant squared portion of the pattern, this angle is the lower limit of the cosecant squared portion and is always positive. MAX-EL-FOR-CSC2 is the upper limit of the cosecant squared portion of the antenna pattern. At elevation angles greater than this angle the antenna's gain is again the minimum gain.

The angular range within which the cosecant squared antenna gives constant returns from targets moving at constant heights is quite small. It is encompassed in azimuth by AZ-BEAMWIDTH and its elevation range is bordered by PEAK/CSC2-BOUNDARY-EL and MAX-EL-FOR-CSC2.

In principle the ANTGR-PATTERN command may be used for automatic generation of many precompiled patterns from real radar systems, (these are, it seems, described in a US secret classified document [1]). Its format is:

```
ANTGR-PATTERN
  PATTERN-ID          13.0  (NO-UNITS)
  GAIN-CORRECTION      0.0  (DB)
  MIN-GAIN           -10.0  (DB)
  EL-ROTATION          0.0  (DEG)
END ANTGR-PATTERN
```

Here the PATTERN-ID is the label of the precompiled pattern, it may be one of 13, 14, 16, 18-26, 28-30, 39, 41-52, 54-59, 62-64, 74, 79-87, 90-94, 100-127 and 131-133. These values must be given as a real number with a decimal point. In practice, since we don't know which patterns correspond to which radars this generating function is not so useful. Each pattern may be modified from the values in [1] by using GAIN-CORRECTION, MIN-GAIN and EL-ROTATION. The GAIN-CORRECTION changes the peak gain of the antenna by the specified amount, and MIN-GAIN will supersede the tabulated minimum gain if it is larger than this value. Finally EL-ROTATION may be used to tip the pattern up and down relative to the tilt angle given in [1].

Two entries for sensor transmitters indicate how well they may be detected by warning receivers. The first is the strength of the antenna's backlobe, which is susceptible to detection by a warning receiver. It may be set with the MIN-GAIN item:

```
MIN-GAIN -10.0 (DB)
```

A synonym, RWR-BACKLOBE-GAIN, also exists for this item. Furthermore, if a sensor transmitter may be detected by a warning receiver then the INTERCEPT-INTERACT data item must be set, e.g.

```
INTERCEPT-INTERACT
  seeker_rx  warning_rx
END INTERCEPT-INTERACT
```

where 'seeker_rx' and 'warning_rx' are the names of two warning receivers which may detect the transmitter. Furthermore, the susceptibility definition of the element to which the sensor transmitter is a part should include the entry `SNR-ELE-INTERACTIONS` which also must name the relevant radar warning sensor receivers.

Suppressor does not model all the detailed physics of radar receivers but the setting of the transmitter's properties should be consistent with those of a real system. As an example of this consider how pulsed radars compute the range and velocity of targets, a capability that Suppressor models implicitly. The range of a target may be found by measuring the round trip travel time of the pulse. For example, a target at a distance R will return an echo to the receiver after a time $2R / c$, where c is the speed of light. However, if the pulse cycle length is T, then another target at distance some $cT / 2$ greater will return the previous pulse at the same time, and the receiver will be unable to distinguish between these targets. While several techniques exist to discriminate these ranges, such as tagging the individual pulses or varying the pulse repetition frequency (PRF) so that T changes, a simple method is to choose the PRF to match the radar's expected range. This means that the returns from targets at larger ranges will never be detected because they are too weak and hence the range of the target can be deduced with some confidence. So if a radar has a range R_d of 200 km we would wish that this is equal to $cT / 2$, and so $T = 2R_d / c = 1 / 750\,\mathrm{s}$ or a PRF of 750 s^{-1}. In general, with this criterion long radar ranges match low PRFs and vice-versa. Now Suppressor models none of the above, but will instead allow the user to set arbitrary PRFs and detection ranges and still accurately obtain the range of the target. This should be borne in mind when radar properties are modelled to ensure that the modelled radar does not perform with impossibly high certitude.

Another subtlety of pulsed radars is associated with the width of the pulse used by the transmitter. The relative accuracy of the measured range depends on the width of the pulse. If the pulse is spread over a large fraction of the whole cycle then the measured range will be very uncertain. If, on the other hand, the pulse is very short then the returned range will be quite accurate. In addition using long pulses causes problems with 'eclipsing', which is the loss of returned echoes which reach the dish when the radar is transmitting. On the other hand, as we shall see when we discuss the maximum range of a radar, the longer the pulse the greater the detection range of the radar. These mutually conflicting aims can be partly reconciled by a technique known as pulse compression. Here relatively long pulses are modulated, often by varying their frequency, so that when the echoes are received a compressed pulse can be reconstructed electronically. This compressed pulse, being much shorter than the outgoing pulse provides much more accurate range information than an uncompressed pulse. This technique does not of course reduce problems due to eclipsing, but does allow quite high average power outputs to be obtained from the transmitter. Suppressor sets these characteristics using the keywords `PULSE-COMPRESSION-RATIO` and `DUTY-CYCLE`, where `DUTY-CYCLE` actually refers to the length of the compressed pulse not the uncompressed pulse as might have been expected. For example:

```
PULSE-REPETITION-FREQ      750.0     (HZ)
DUTY-CYCLE                   2.0E-3  (NO-UNITS)
PULSE-COMPRESSION-RATIO    100.0     (DB)
```

sets the uncompressed pulse to be one fifth (0.2) of the total cycle of $1/750$ s, with a compressed pulse with a width of only 2.7 μs. Again Suppressor will provide accurate radar ranges independently of the setting of the PRF and the duty-cycle, and this should be remembered when modelling sophisticated modern radars using Suppressor.

If a sensor transmitter has the capability to change frequencies to avoid jamming then the time it takes for this to occur is set with `CHANGE-FREQUENCY-DELAY`. This option is only available if alternative frequencies are defined in the SDB `ALT-FREQ:` entry and if the linked receiver has the capability to recognise jamming.

The functioning of a radar depends not only on the sensor transmitter but also on the sensor receiver. The capabilities of a sensor receiver are determined in a separate data block to the transmitter using Suppressor. However, several of the options are common to both transmitters and receivers. For example, the antenna pattern of the sensor receiver must also be set. In practice since most radars use the same antenna for both transmission and reception, these patterns will normally be identical for paired receivers and transmitters. Other data items which may be set for both receivers and transmitters include the `PEAK-GAIN`, the `ELEV-SLEW-LIMITS`, the `INTERNAL-LOSS` and the `VERTICAL-OFFSET`. Of these `PEAK-GAIN` and `INTERNAL-LOSS` specify the gain at the centre of the main lobe and the efficiency of the sensor transmitter and receiver in transmitting and receiving radiant energy respectively. The remaining options refer to the physical characteristics of the radar installation. The `ELEV-SLEW-LIMITS` gives the range of elevations in which the system may be pointed. So that a system with the settings:

```
ELEV-SLEW-LIMITS
  MAX-EL     75.0  (DEG)
  MIN-EL    -25.0  (DEG)
END ELEV-SLEW-LIMITS
```

would be able to slew in azimuth from 25° below its reference elevation to 75° above it. These limits restrict only the direction in which the antenna may point and not the directions in which targets may be detected, (since the beamwidth of the antenna is presumably not zero). The azimuthal slew limits may be similarly set, for example:

```
AZIMUTH-SLEW-LIMITS
  LEFT-AZ      35.0  (DEG)
  RIGHT-AZ     65.0  (DEG)
END ELEV-SLEW-LIMITS
```

Note here that both angles must be entered as positive numbers between 0° and 180°. In both cases the reference angles around which these limits are defined may be varied. Normally this angle is due east, but can be set to be the heading of the current location or some other arbitrary direction.

The VERTICAL-OFFSET indicates the physical height of the system above the element on which it is based. This item is relevant when the element is located on the ground, since otherwise a height of zero will be taken and so the apparent radar horizon will be very close indeed to the system. So a radar on a 10 m tall pedestal could be defined as:

```
VERTICAL-OFFSET  10.0 (M)
```

Many more options than those discussed so far are used in defining the properties of sensor receivers and although not used explicitly in the transmitter's capability block several of them influence the behaviour of the sensor transmitter too.

3.4.4.2 Sensor Receivers

Several data items are relevant to all four classes of sensor receiver, and these will be discussed first. The first two items that would be present in most sensor receiver capability blocks are the SNR-CHARACTERISTICS table and the QUALITY-OF-DATA block. The first of these describes the basic characteristics of the receiver and the second the types of information that may be deduced from the receiver.

First consider an example sensor receiver's characteristics, '1030 aew_radar_rx':

```
CAPABILITY aew_radar_rx_data
  SNR-CHARACTERISTICS
    PHYSICAL-SCAN  ACQ
    IFF            RADAR
  END SNR-CHARACTERISTICS
  <Omitted capabilities>
END CAPABILITY
```

There are six options which may be set to define the sensor receiver's characteristics. If omitted five of the six options will take default values. If the sixth option is omitted, which sets the receiver's polarization this would simply mean that the sensor detects unpolarized emissions.

The first option, which may be either PHYSICAL-SCAN or FREQ-DRIVEN, determines how Suppressor computes sensing chances. Unless the POINT IT entry in the SDB is used to specify a fixed point or a particular target at which the antenna is pointed Suppressor always assumes that a sensor receiver's beam is rotating. It does not model this rotation in detail though, but instead assumes the radar illuminates the whole of its surroundings at regular intervals. These intervals are specified by a SENSING-MODE-RATES entry and would correspond to the rotational period of the radar.

At these instances the antenna is assumed to be pointing as much as physically possible directly at the target. If the default option PHYSICAL-SCAN is chosen then the antenna must always point horizontally, i.e. at zero azimuth. Therefore the target may be somewhat above or below where the antenna is pointing and so will not be detected with the antenna's mainbeam. If the option chosen is FREQ-DRIVEN then in the absence of other constraints the sensor receiver will point directly at the target.

The consequences of the above for radar systems are that normally a FREQ-DRIVEN radar will always compute a sensing chance with the antenna's peak gain. A PHYSICAL-SCAN radar will however use at least the elevation information in the ANTENNA-PATTERN. Hence a highly accurate ANTENNA-PATTERN not required very often.

The above pointing behaviour can be modified in several ways. As noted above the SDB command POINT IT may be used to make the antenna point at a certain position or in a certain direction. The slew limits of the sensor described above, i.e. AZIMUTH-SLEW-LIMITS and ELEV-SLEW-LIMITS may be invoked and these could also prevent the sensor pointing directly at the target. In this case detections can still be made beyond these slew limits since the sensor will have a field of view extending beyond these limits. (Although the sensor's sensitivity may not be the same in all directions.) Note that a FREQ-DRIVEN sensor with elevation slew limits of 0° would in effect be exactly like a PHASE-DRIVEN system. Furthermore, ELEV-SLEW-LIMITS have no effect on PHYSICAL-SCAN sensor receivers since these may not slew vertically in any case.

A limiting field of view of the sensor can be set with the SNR-ANGULAR-LIMITS item. Targets which lie outside the field of view of the sensor cannot be seen. Symmetric limits may be set, for example:

```
SNR-ANGULAR-LIMITS
  AZ-LIMIT 180.0 (DEG)
  EL-LIMIT   5.0 (DEG)
END SNR-ANGULAR-LIMITS
```

which would limit the field of view of the sensor receiver to a narrow slot of plus or minus 5° from the reference elevation whilst not restricting the azimuthal range at all. Asymmetric limits may also be set, as in:

```
SNR-ANGULAR-LIMITS
  LOWER-AZ-LIMIT  -45.0 (DEG)
  UPPER-AZ-LIMIT   37.5 (DEG)
END SNR-ANGULAR-LIMITS
```

which, again, would only limit the azimuthal view angle. Of course these limits are completely ineffectual for FREQ-DRIVEN systems if no constraints are placed on the direction in which the antenna may point, and furthermore the ANTENNA-PATTERN command or its alternatives could provide the same functionality.

The meaning of the above angular ranges depends on the setting of the reference vector about which they are defined. As noted previously this reference vector is itself variable and may be determined in several different ways. In order of precedence these are:

- an explicit POINT-IT option specified in the SDB description of the sensor receiver system,
- the velocity vector of the current location,
- the HDG: element in the SDB location description (which may apply even to a stationary location),
- the default setting of due east.

Returning now to the SNR-CHARACTERISTICS entry sensor receivers may be used either for acquisition or tracking of targets. By default the receiver is assumed to be for acquisition only, as indicated by the ACQ keyword. Alternatively, the receiver may be defined to be for tracking purposes only, indicated by TRK, or for both purposes, shown by BOTH-ACQ/TRK. In certain circumstances Suppressor can use tracking only sensors for acquisition purposes. This may occur if all the acquisition radars possessed by the player are destroyed and will be recorded with the output message 'USING-TRACKER-FOR-ACQ-ALSO' which is described in the reference manual.

When a receiver has the capability to both acquire and track targets it may be able to still acquire new targets whilst tracking. This ability may be set with the (default) option ACQ-WHEN-. Alternatively, it may not be able to do so, which must be set using the NO-ACQ-WHEN-TRK keyword.

The default IFF setting indicates that the receiver can distinguish the 'friend-or-foe' status of the target. With the alternative NO-IFF setting the receiver cannot ascertain this information.

Another option describes which of the four varieties of sensor receiver this sensor is an example of. Sensor receivers, as noted above, may be a part of an active radar system or passive warning receivers, or function in the optical or infrared bands. The default condition is that the receiver is a radar, as indicated by the RADAR item. The other options, corresponding to the other types of sensor receiver, are WARNING-RCVR, OPTICAL and INFRA-RED respectively.

A final optional setting determines the polarization of a sensor receiver. It is only appropriate for radars and warning receivers, and then only if the radar cross-section of the targets are defined for polarized signals. The polarization may be one of HORIZ-POL, VERT-POL, LEFT-CIR-POL and RIGHT-CIR-POL. It has no default setting, but when omitted the receiver will function with unpolarized signals only.

In addition to SNR-CHARACTERISTICS a further qualitative description of the receiver is given by the QUALITY-OF-DATA block. This indicates what information Suppressor can collect from each receiver. At the very least the two entries ALT (altitude) and PLANAR-LOCATION are necessary for Suppressor to operate correctly. This implies that Suppressor will always be able to deduce the range of a target, regardless of the physical realities of the system being modelled. In addition if tactical decisions are to made based on information from the receiver (almost invariably the case), then TYPE-OF-ELEMENT and TYPE-OF-PLAYER must be included. This allows Suppressor to figure out what the items being sensed are and so make sensible tactical decisions regarding them. Often all eight possible entries are included, as is the case in the example below:

```
QUALITY-OF-DATA
  TYPE-OF-ELEMENT  AZ   ALT      TYPE-OF-PLAYER
  PLANAR-LOCATION  SPD  HEADING  NO-OF-ELEMENTS
END QUALITY-OF-DATA
```

If, for example, SPD were omitted the receiver would not be able to determine the speed of the target. This may be the only way to mimic the capabilities of some radars in certain circumstances. Furthermore, some sensor systems, for example optical sensors, are unlikely to give accurate speed and heading information and so these entries could in these cases be omitted.

The sensitivity of the sensor receiver is given by the DETECTION-SENSITIVITIES item. The options that may be set within this block vary in detail from one type of sensor receiver to another, but two items may be set for all types of sensor receiver.

```
DETECTION-SENSITIVITIES
  SENSING-THRESHOLD         3.0  (DB)
  POST-LOCKON-THRESHOLD   -10.0  (DB)
END DETECTION-SENSITIVITIES
```

This example is for a radar receiver, in which the detection thresholds are signal-to-noise ratios. For optical and infrared sensors these same thresholds are probabilities and so are not expressed in decibels. The SENSING-THRESHOLD (or its synonym S/N-THRESHOLD) is the critical value at which a target is just detectable by the receiver in normal operation. (Which is to say acquisition mode or initial tracking.) Once a target is already locked on to by a tracking sensor a lower threshold, the POST-LOCKON-THRESHOLD can be used.

The rate at which the receiver illuminates the target, (which mimics the sweep rate of this receiver), is set in the `SENSING-MODE-RATES` entry:

```
SENSING-MODE-RATES
  ACQ-SENSING-RATE         0.1  (1/SEC)
END SENSING-MODE-RATES
```

The example receiver has acquisition capability only, so only a `ACQ-SENSING-RATE` is set, to 0.1 s^{-1} (so that the rotation period of this radar is 10 s). A more capable system may have up to four different values set, for example:

```
SENSING-MODE-RATES
  ACQ-SENSING-RATE         0.1  (1/SEC)
  TRACK-SENSING-RATE       0.5  (1/SEC)
  FIRING-SENSING-RATE      2.0  (1/SEC)
  REACQUISITION-TIME      20.0  (SEC)
END SENSING-MODE-RATES
```

The `TRACK-SENSING-RATE` and `FIRING-SENSING-RATE` indicate the frequency at which the target's position will be monitored when the receiver is tracking, and being used for weapon guidance respectively. These rates are much higher than the search rate used in acquisition. If the receiver is an optical sensor then if a tracked target is lost a time specified by the `REACQUISITION-TIME` may be spent looking for the target in the vicinity of where the target was last seen. After this time is spent the receiver reverts to acquisition mode. If this time is not specified the reciprocal of the `ACQ-SENSING-RATE` is used.

In addition to the sensing rates several time-delays can be defined for sensor receivers. These effect how quickly these sensors can interact with a thinker. The delays are given using the `SNR-TIME-DELAYS` item:

```
SNR-TIME-DELAYS
  START-LOCKON-DELAY        10.0  (SEC)
  MAX-COAST-TIME             7.0  (SEC)
  POST-LOCKON-S/N-DELAY      4.0  (SEC)
END SNR-TIME-DELAYS
```

If this entry is omitted all these values will be set to zero. The first delay, `START-LOCKON-DELAY`, is the interval between the thinker starting to use a tracking sensor and the first possible sensing chance. The `MAX-COAST-TIME` has two uses within Suppressor: the first is the maximum time a tracker can remain in coast mode after losing a tracked target without aborting any fired ordnance. So with the default setting of zero any lost track will immediately abort any firing sequence. For a setting of `MAX-COAST-TIME` to be effective it should allow at least one more sensing opportunity, and so be larger than the reciprocal of both the firing and tracking sensing rates. A second use of `MAX-COAST-TIME` is to determine whether or not a lost target can be recovered with a single successful 'hit', i.e. a single detection. If the target is re-detected within this period only one hit will suffice to re-establish the track, otherwise several hits, as set by the HITS-TO-TRACK item, will be required. In this case the default value is not zero but the TIME-TO-DROP value of the thinker linked to the sensor. This interpretation of MAX-COAST-TIME applies to all sensors, even those with only an acquisition capability. The final delay applies only to tracking sensors and determines how long the SENSING-THRESHOLD for the detection signal to noise ratio is used after tracking commences. Once this time is expired the usually lower POST-LOCKON-THRESHOLD may be used. Again this delay should be large enough for sensing chances to have actually occurred, i.e. it should be larger than the reciprocal of the TRACK-SENSING-RATE.

An important item for all sensor receivers is the setting of the number of successful detections (hits) that a receiver needs to make in identifying a target. This is specified by the HITS-TO-ESTABLISH-TRACK item, (where TRACK here applies even to receivers which have a target acquisition mode only):

```
HITS-TO-ESTABLISH-TRACK  4 (NO-UNITS)
```

In this case as the receiver sweeps around four detections must be made to allow the player's thinker to perceive the target. If this item is not present the default value is unity. HITS-TO-ESTABLISH-TRACK is a synonym for the keyword HITS-TO-ESTABLISH-TRACK-(DRY), and both specify the number of hits required in the absence of jamming. A restriction may be made on the number of sweeps of the sensor within which the required number of hits must occur using the following:

```
SCANS-IN-ESTABLISHING-TRACK-(DRY)  6 (NO-UNITS)
```

this would require the four hits be made within six consecutive scans. If this item is absent no restriction is made on the number of scans. Both these items may be modified in the presence of jamming, so for example:

```
SCANS-IN-ESTABLISHING-TRACK-(JAM)  8 (NO-UNITS)
HITS-TO-ESTABLISH-TRACK-(JAM)     6 (NO-UNITS)
```

would mean that when jamming is present six scans out of eight must detect the target for the target to be perceived.

The limit on the number of scans should be less than the number of bits in an integer in the computer's version of FORTRAN, (remember Suppressor is encoded using FORTRAN). This is due to the implementation of the algorithm for checking the number of hits made in the set number of scans, which presumably counts bits in an integer variable. For most compilers this limits SCANS-IN-ESTABLISHING-TRACK to be 31 or less.

The number of targets that an individual tracking sensor receiver is able to follow simultaneously can also be set using:

```
MAX-PARALLEL-TRACKS  4
```

with the default value being unity. This value is the maximum possible number of targets that can be tracked, and may be reduced as a consequence of the operational conditions.

A range-altitude envelope for the sensor receiver is required by Suppressor. This acts as a filter, only allowing targets which are within the filter to have sensing chances computed by Suppressor. (More precisely only targets within the envelope will be eligible for inclusion on the sensing chance list, SCL, discussed in section 2). The envelope is defined by the RNG-ALT-CAPABILITY entry:

```
RNG-ALT-CAPABILITY
  DIMENSION 1 RNG (KM)  0.0  300.0
    MIN-ALT (KM) MAX-ALT (KM)
      -10.0      10.0
END RNG-ALT-CAPABILITY
```

and is an approximation of the range versus elevation capability of the sensor receiver. The example table is of a simple envelope that only includes targets that lie within a 300 km ground range and a vertical range of plus or minus 10 km. A more complicated table could be defined with the help of extra range intervals to better approximate the receiver's antenna pattern.

Errors in the perceived position of targets detected by a FREQ-DRIVEN sensor receiver may be modelled with the SEEKER-ERROR-DATA item. The measured azimuth and elevation of a target are related to the true azimuth and elevation by two pairs of coefficients. The first of these pairs determines the orientation of the antenna's boresight, the second pair determines the measured deviation of the target from the direction of the boresight. For example, the true azimuth Φ of a target is related to the azimuth of the boresight, Φ_B and the azimuthal deviation of the target $\Delta\Phi$ by:

$$\Phi = \Phi_B + \Delta\Phi .$$

The measured azimuth $\tilde{\Phi}$ is given by:

$$\tilde{\Phi} = \varepsilon_B + \Phi_B + \varepsilon_S \Delta\Phi ,$$

where the azimuthal errors are the boresight error, ε_B , and the scale error, ε_S , respectively. These errors are modelled in Suppressor as being drawn from a Gaussian distribution whose mean and standard deviation are specified by the user in the SEEKER-ERROR-DATA table. For the azimuthal errors the keywords are AZ-BORESGT-ERROR-MEAN and AZ-BORESGT-ERROR-DEV for the boresight error, and AZ-SCALE-ERROR-MEAN and AZ-SCALE-ERROR-DEV for the scale error. The errors in elevation are treated in the same manner. An example is:

```
SEEKER-ERROR-DATA
  AZ-BORESGT-ERROR-DEV    1.0  (DEG)
  AZ-BORESGT-ERROR-MEAN   0.3  (DEG)
  EL-BORESGT-ERROR-DEV    0.5  (DEG)
  EL-BORESGT-ERROR-MEAN   0.4  (DEG)
  AZ-SCALE-ERROR-DEV        0.2  (DEG)
  AZ-SCALE-ERROR-MEAN       1.0  (DEG)
  EL-SCALE-ERROR-DEV        0.4  (NO-UNITS)
  EL-SCALE-ERROR-MEAN       0.9  (NO-UNITS)
END SEEKER-ERROR-DATA
```

If this feature is required for a PHYSICAL-SCAN radar (which cannot tilt out of its azimuthal plane of rotation) then a PHYSICAL-SCAN radar may be emulated with the help of the ELEV-SLEW-LIMITS item as follows:

```
SNR-CHARACTERISTICS
  FREQ-DRIVEN
END SNR-CHARACTERISTICS
ELEV-SLEW-LIMITS
  MIN-EL  0.0  (DEG)
  MAX-EL  0.0  (DEG)
END ELEV-SLEW-LIMITS
```

Random uncertainties can also be used to effect the performance of tracking sensor receivers. These receivers will be in one of two states: either be trying to lock on to a target or trying to maintain a lock that was already established. The transition from one state to the other can be partially randomized with the help of the SNR-TRACKING-PROBABILITIES entry. So that the table:

```
SNR-TRACKING-PROBABILITIES
  INITIAL-LOCK-PROB    0.65  (NO-UNITS)
  CONTINUE-LOCK-PROB   0.95  (NO-UNITS)
END SNR-TRACK-PROBABILITIES
```

would imply that once all other criteria were met there is only a 65% chance that a lock on will result. Furthermore, at any time there is a 5% chance that an established target lock could be lost due to random factors.

In later sections the interactions of a radar receiver with explicitly modelled electromagnetic jamming will be discussed. However, will be noted in the discussion on DISRUPTOR systems, Suppressor can model disrupting implicitly. These disruptors are modelled with considerably less fidelity than explicit disruptors, but can represent the effectiveness of disruptive counter-measures in probabilistic terms. Two entries are relevant here: IMPLICIT-CM-INTERACT which lists the names of the implicit disruptors which can interact with a sensor receiver, and EFF-BURST-CM-PROB which determines the probability that any of the implicitly modelled countermeasures are effective. Suppose that a receiver includes the entry:

```
IMPLICIT-CM-INTERACT
  chaff  flare
END IMPLICIT-CM-INTERACT
```

then either of the two implicitly modelled disruptors 'chaff' and 'flare' may interfere with the receiver. (In this case we might expect the receiver to be an optical or infrared sensor.) Should either of these disruptors be operating at the same location as the target then an otherwise successful sensing chance may be disrupted. So should the SENSING-THRESHOLD be satisfied for the receiver Suppressor would determine the probability that either disruptor blocks the sensing chance with the entry of the form:

```
EFF-BURST-CM-PROB 0.95 (NO-UNITS)
```

In this case there would be a 95% chance that the sensing chance would be disrupted. Note that the effectiveness of different implicit disruptors against the current sensor receiver cannot be distinguished using Suppressor. Instead a single effectiveness must be used for all implicit disruptors with the potential to interfere with the receiver.

Radar Sensor Receivers and Warning Sensor Receivers

Radar sensor receivers and warning receivers form a very important class of systems in Suppressor. Both of these systems operate in the radio frequency bands. Warning receivers differ from radar receivers only because they are detect signals from other players' radar or radio systems rather than echoes from a linked sensor transmitter. Many of the CAPABILITY block items are exclusive to one or both of these types of receivers, and these are discussed in the following paragraphs. In what follows the two classes of system are referred to generically as radar receivers. When a warning receiver or radar sensor receiver is specifically meant this will be made clear in the text.

Just as with sensor transmitters the ANTENNA-PATTERN command may be used to determine the response of the radar receiver's antenna to signals arriving from various directions. As was noted above, in most cases detections are made with the mainbeam of the radar but the sidelobes will be used when the mainbeam cannot directly illuminate the target. Note that the ANTENNA-PATTERN command should *not* be used to define antennae with side-looking mainbeams. Because Suppressor assumes the antenna is pointing directly at the target it takes the gain at an azimuth 0° in computing the sensing chance instead of the peak gain. If a side looking radar is required then an aligned ANTENNA-PATTERN should be defined and then the orientation of the antenna should be rotated by 90° using the SDB command POINT IT:

```
POINT IT IN REL DIR AZ,EL:   0.0  90.0  (DEG)  FIXED
```

which means that the antenna will always point at an angle 90° away from heading of the element carrying the radar.

If the radar is of the PHYSICAL-SCAN type then the ANTENNA-PATTERN may be used to skew the mainbeam up or down in elevation. For example, the data block:

```
ANTENNA-PATTERN
  DIMENSION 1 AZ   (DEG)   0.0 180.0
    DIMENSION 2 EL (DEG) -90.0  10.0  30.0  90.0
      GAIN          (DB)     -10.0  30.0 -10.0
END ANTENNA-PATTERN
```

would define an antenna with a beamwidth of 20° in elevation and a mainbeam skewed upwards by 20°. This antenna pattern could be used for a radar which whilst rotating in a plane parallel to the ground has a mainbeam which points upwards.

The DETECTION-SENSITIVITIES item, which defines the ability of the sensor to detect incoming signals has been briefly mentioned above. We noted that two thresholds, one for sensing and one used once lock-on has occurred exist for sensor receivers. In the case of radar receivers the thresholds are measured in decibels and are logarithms of the minimum signal to noise ratios which just allow a detection to be made. So a SENSING-THRESHOLD of 3.0 dB means that any signal should be approximately twice the strength of the total noise to be discernible.

The noise which is inherent to the detector and other losses which occur during the passage of the signal through the detector can be set using the DETECTION-SENSITIVITIES command. For all radar receivers, including radar warning receivers the internal noise and the internal losses can be specified. The internal noise is in reality a noise power but can be expressed in decibels as a ratio by expressing it in terms of a reference power of 1 W. In this way a RECEIVER-NOISE figure of -100 dB would mean that the receiver radiates noise at a rate of 10^{-10} W. The internal losses are similarly expressed in decibels but are given in terms of the efficiency of the receiver.

(In other words the Suppressor keyword INTERNAL-LOSSES is a misnomer.) Hence a setting of INTERNAL-LOSS as -10 dB means not that the losses are 10% but that the efficiency of the system is 10%, so the losses are actually 90%. For radar sensor receivers alone the OPERATING-LOSSES may also be given. Although similar to internal losses these arise from the functioning of the radar as a whole due to the inefficiencies inherent in matching the transmitted and received signals. As such they do not occur with warning receivers, which have no transmitter with which to cooperate. Furthermore OPERATING-LOSSES will not weaken a signal originating from any other system than the coupled transmitter, so they will not dilute the incoming signal from an explicitly modelled jammer. If omitted the default value for the operating losses is zero. One cause of operating losses is the eclipsing of reflected signals which occurs when the transmitter is operating. This eclipsing loss will be given by the (true) duty cycle of the transmitter linked to the receiver, i.e. the length of the uncompressed pulse.

Relatively few options are restricted to warning-receivers alone, although the converse is certainly not true. One item that is unique to warning receivers is RCVR-FREQ-LIMITS which indicates the range of frequencies a warning receiver can detect. In its absence the receiver will be able to detect any portion of the spectrum. So, for example, setting:

```
RCVR-FREQ-LIMITS
  LOWER-FREQ-LIMIT  1.0 (GHZ)
  UPPER-FREQ-LIMIT 100.0 (GHZ)
END RCVR-FREQ-LIMITS
```

would cover a very wide range of frequencies including the emissions from most radar systems.

Another option that is restricted to warning receivers is the RWR-DETECTION-CRITERIA. This criterion may be set to one of three options which indicates whether or not a receiver has the capability to detect emissions from a radar transmitter's backlobe beam. The possible options are:

- MAINBEAM-ONLY: the default which indicates that only the transmitter's mainbeam can be detected,
- BACKLOBE-ONLY: where only the transmitter's backlobe can be detected,
- BOTH-MAINBEAM/BACKLOBE: where both the mainbeam and the backlobe can be detected.

Note that warning receivers has two potential sensing chances for every one of a radar receiver with the same SENSING-MODE-RATES, one with the mainbeam and one with the backlobe.

If a warning receiver is able to detect targets using backlobe emissions the capability HITS-TO-ESTABLISH-TRACK-(BL) must be given. This indicates the number of backlobe hits needed to track a target. Unlike the mainbeam, where if the corresponding keyword `HITS-TO-ESTABLISH-TRACK-(MB)` has a default value of one if omitted, leaving out this item for the backlobe implies that the backlobe hits will be ignored and no track formed. (`HITS-TO-ESTABLISH-TRACK-(MB)` is yet another synonym of `HITS-TO-ESTABLISH-TRACK`.)

Note that the detection of a radar transmitter by a warning receiver depends not only on the direction in which the warning receiver's antenna is pointing, but also on the orientation of the radar transmitter's antenna. In the absence of other constraints Suppressor assumes that for a mainbeam sensing chance the transmitter is pointed directly towards the warning receiver, whilst for a backlobe sensing chance it is pointed directly away. If, however, the radar transmitter is tracking a target it will instead be assumed to be pointing at that target and so a sidelobe of the transmitter will be used in computing the sensing chance. The same will be true if either of the two radars' orientations are constrained by a `POINT IT` entry in the SDB. Finally if either antenna has physical slew limits set then the radars will be assumed to be pointing as nearly as possible directly towards or away from each other.

For radar sensor receivers the nominal range of the radar system may be determined from the various capabilities of the receiver and its linked transmitter. Suppose that a radar transmitter is emitting a signal which is reflected from a target at a distance R from the transmitter. The transmitter's power is P and its antenna has a gain G_{TX}. Then if the internal losses of the transmitter are L_{TX} the signal power density S at the target is:

$$S = \frac{G_{TX}\, P}{4\pi\, L_{TX}\, R^2}.$$

If the target has a radar cross-section of σ then the reflected signal, after travelling back a distance R to the receiver, will have power density:

$$S = \frac{G_{TX}\, P}{(4\pi)^2\, L_{TX}\, R^4}\sigma.$$

Both of the above relationships arise quite simply from the inverse square law for radiation and the definition of antenna gains in terms of effective antenna areas.

The detected signal must be intercepted by the receiver's antenna, which has a gain G_{RX} at the signal's wavelength of λ, and so an effective area of:

$$G_{RX}\lambda^2 / 4\pi .$$

Since the internal losses in the receiver are L_{RX} the detected signal will have power:

$$S = \frac{G_{RX}\, G_{TX}\, P}{(4\pi)^3\, L_{RX}\, L_{TX}\, R^4}\sigma\lambda^2 .$$

Note that almost always the same antenna is used for both transmission and reception, implying that G_{RX} must equal G_{TX}.

The receiver works by integrating the power received over some time T. The noise power collects uniformly over this interval, but the signal energy, as discussed in the sensor transmitter section, occurs in a pulse of duration τ. Therefore the resulting signal to noise ratio S/N is:

$$\frac{S}{N} = \frac{G_{RX}\, G_{TX}\, P}{(4\pi)^3\, L_{RX}\, L_{TX}\, R^4}\frac{\delta\sigma\lambda^2}{L_{OP} N},$$

where $\delta = \tau / T$ is the duty cycle of the transmitter. The term L_{OP} represents the operating losses, and if taken to be due solely to eclipsing will be equal to the reciprocal of the complement of the duty cycle, $(1-\delta)^{-1}$. It is now possible to compute the reference range of a sensor radar receiver in Suppressor. The `RECEIVER-NOISE` is a constant C which is equal to the noise power N multiplied by the scaling term of 1 W. The critical sensing signal to noise ratio S_R is set by `SENSING-THRESHOLD`, and the target's cross-section σ is taken to be 1 m^2. Hence the reference range R_d is given as:

$$R_d^4 = \frac{G_{RX}\, G_{TX}\, \delta\, \lambda^2}{(4\pi)^3\, L_{OP}\, L_{RX}\, L_{TX}\, S_R}\frac{P}{C}.$$

When pulse compression is being used Suppressor interprets the `DUTY-CYCLE`, *d*, not as the fractional amount of time the transmitter is on, but rather as the compressed pulse width. Hence $\delta = d\, p$ where d is the compressed pulse width and p is the degree of pulse compression. Furthermore, note the values set for all the losses are in fact their reciprocal transmittances, so, for example, setting `OPERATING-LOSS` to -1 dB implies a true loss L of 1 dB.

Calibration of a radar's performance to meet a desired range can be achieved by adjusting the noise figure C. This can be done explicitly, so for an example radar receiver we may set the `DETECTION-SENSITIVITIES` as:

```
DETECTION-SENSITIVITIES
  SENSING-THRESHOLD        3.0  (DB)
  RECEIVER-NOISE        -160.0  (DB)
  INTERNAL-LOSSES        -10.0  (DB)
  OPERATING-LOSSES        -1.0  (DB)
END DETECTION-SENSITIVITIES
```

This receiver's paired transmitter shares the same antenna and consequential gains, the remaining capabilities being:

```
PEAK-POWER-OUTPUT             5.0E6    (WATTS)
XMIT-FREQ                     3.0E9    (HZ)
INTERNAL-LOSS               -10.0      (DB)
DUTY-CYCLE                    2.0E-3   (NO-UNITS)
PULSE-COMPRESSION-RATIO     100.0      (DB)
```

These quantities are sufficient to compute the radar's reference range, since we have overall gains of 20 dB in both transmission and reception, and a power output of 5 MW transmitted for 20% of the time at a wavelength of 0.1 cm. The noise output of the receiver is 10^{-16} W which implies a one square metre target will reflect enough energy to meet the sensing threshold of 3 dB at a range R_d given by:

$$R_d^4 = \frac{100 \cdot 100 \cdot 0.1^2 \cdot 0.2}{(4\pi)^3 10^{0.1} 10^{0.3}} \cdot \frac{5 \times 10^6}{10^{-16}} = 2.0 \cdot 10^{20}\ m^4 \ ,$$

which implies that the radar's maximum range is R_d = 119.0 km. This range will be printed out (in units of metres) by Suppressor in the SDB listing file after processing of that stage, it is given by `MODEL MESSAGE 99`.

An alternative mechanism is to set the noise figure internally by using the calibration option `ONE-M2-DETECT-RNG` which specifies the range at which a 1 m^2 radar target can just be detected. So if extra commands shown below are given in the radar's capability block:

```
PEAK-GAIN                   30.0  (DB)
ONE-M2-DETECT-RNG          200.0  (KM)
```

and if the same `PEAK-GAIN` were also included in the transmitter's capability block then these settings will be used to replace the original noise figure of -160 dB. (The `PEAK-GAIN` should equal the mainlobe gain of the `ANTENNA-PATTERN`). The original range was 119 km but the required range is now 200 km, implying that the range is increased by this factor and so the noise figure would be reduced by the fourth power of this ratio, i.e. multiplied by a factor $(0.6)^4$. Hence the noise figure would become -169 dB rather than -160 dB. When this replacement is made internally the new noise figure is printed out in the SDB stage listing file as `MODEL MESSAGE 98`.

Note that it would be better if the range implied by the noise figure and the detection range agree, so that no confusion can arise over the true range of the radar. Therefore care should be taken that the relevant settings are made rather than relying on Suppressor to calibrate the system automatically.

The actual range at which targets may be seen depends additionally on the radar cross section of the target, albeit only weakly since it is dependent on the fourth root of the cross-section. So increasing the radar cross section by a factor of hundred will increase the range by $\sqrt{10}$, only slightly more than a factor of three. Targets which are within the nominal range of the radar might still not be seen if they are hidden by the curvature of the Earth or by local topographic features, such as a hill for example. Suppressor is capable of checking both of these items. The Earth's curvature is represented by the `EFFECTIVE-EARTH-RADIUS` item. It will generally be larger than the true radius of the Earth to account for atmospheric refraction of the signal. For radio frequencies (as used by radars) an amplification of 33% is conventionally taken, giving:

```
EFFECTIVE-EARTH-RADIUS  8495.0 (KM)
```

Two points to note here are that if this item is not defined the radius is taken to be infinite, so that the scenario is taking place on an Earth without curvature and so without any horizons. Furthermore Suppressor will be happy to accept any value for the radius, so scenarios can be situated on other planets! The distance of the horizon depends on the height of the receiver above the surface, and is approximately $\sqrt{2Rh}$ with effective radius R and height h. A target at height H may be seen at a maximum slant range of $\sqrt{2R}\left(\sqrt{h}+\sqrt{H}\right)$. Hence, as the altitude of the receiver goes to zero the horizon closes in to a point, at least when both the target and the receiver are located on or near the ground. This explains the importance of the `VERTICAL-OFFSET`, which may be used for both sensor receivers and transmitters.

The background noise against which the target is to be detected can be explicitly modelled using a `CLUTTER-TABLE` data item. This table indicates the amount of noise due to clutter as a function of altitude above ground level and range of the target. (The range is that projected long the Earth's surface and not the true three dimensional range of the target.) An example table is:

```
CLUTTER-TABLE
  DIMENSION 1 ALT    (M)
     0.0    10.0
    DIMENSION 2 RNG (KM)
        0.0    300.0
    CLUT-POWER (DB)
        10.0
END CLUTTER-TABLE
```

This table is very simple, containing only clutter noise power for a single pair of intervals. In realistic models more complicated tables would be used.

Since any clutter must degrade the chances of sensing the target the noise, C, present at the receiver is replaced by a total noise defined as $C(1+c)$, where c is the clutter noise. Hence, a receiver noise of -160 dB and clutter noise of 10 dB would give a total noise of -149.6 dB. (Note that the larger the noise figure the noisier the system.) This relationship means that even a clutter noise c of 0 dB would still result in a total noise increase of 3 dB above and beyond the receiver noise.

Various other losses in addition to noise can attenuate electromagnetic signals. We have already discussed internal losses in the transmitter and receiver, but transmission losses through the atmosphere may also occur. These may be specified within Suppressor using the `TRANSMISSION-LOSS` table:

```
TRANSMISSION-LOSS
  DIMENSION 1 FREQ (MHZ)
        450.0  1000.0
    DIMENSION 2 ALT-XMIT (M)
         -1.0  500.0  5000.0
      DIMENSION 3 ALT-RCVR (M)
         -1.0  500.0  5000.0
        DIMENSION 4 2D-DIST (KM)
          0.0  25.0  50.0  100.0  250.0
          GAIN (DB)  -3.0  -10.0 -15.0  -25.0
        DIMENSION 4 2D-DIST (KM)[16]
          0.0  250.0
          GAIN (DB)  0.0
      DIMENSION 3 ALT-RCVR (M)
         -1.0  500.0  5000.0
        DIMENSION 4 2D-DIST (KM)[17]
          0.0   250.0
          GAIN (DB)  0.0
        DIMENSION 4 2D-DIST (KM)
          0.0  250.0
          GAIN (DB)  0.0
END TRANSMISSION-LOSS
```

This table gives gain factors, so a negative gain implies that the signal is weakened by its passage through the atmosphere. Hence, as with the internal losses of receivers and transmitters Suppressor actually treats these losses as efficiencies or transmittances. The transmittances are described in terms of four items: the frequency of the radiation, the altitudes of the transmitter and receiver and the two dimensional ground distance between the transmitter and receiver. The nomenclature of 'transmitter' and 'receiver' is purely notional, since in different circumstances the actual role of these end points may differ. This is not important since the resulting transmittances entered as function of the separation should always be symmetric, i.e. the transmittance between a receiver at an altitude of 10 m and a transmitter at 1 km would be the same as for the receiver at an altitude of 1 km and the transmitter at 10 m. Thus, entering the values for the transmittance table requires some care to ensure that the values are indeed symmetric.

In discussing sensor transmitters the relationship of the PRF to the maximum unambiguous range was remarked upon. It was noted that large unambiguous ranges can be obtained with the help of low PRFs. The selection of an optimal PRF is, as might be expected, rather more complex than such a simple picture might suggest.

[16] This entry should equal the entry labelled 17.
[17] This entry should equal the entry labelled 16.

Suppressor has the capability to model pulsed Doppler radars, which use the differences in the frequency of the returned radar signal to deduce the velocity of the target and so provide a MTI (moving target indicator). In this case stationary targets can be filtered out electronically leaving only moving targets present on the display. Unfortunately the returned signal's frequency depends not only on the relative motion of the target to the radar and the radar's transmission frequency but also on the PRF. This is because the transmission frequency and the PRF beat with each other, which results in the returned spectrum being divided into bands with width equal to the PRF. So the frequency of any two echoes which differ by an integer multiple of the PRF will be indistinguishable. The requirement that the returned Doppler signature be unambiguous is that the PRF is large, which is exactly opposite to that for a large unambiguous range. Since Suppressor will not check the consistency of the capabilities defined for a radar care should be taken in setting both the MTI attenuation table and the sensor's Doppler limits, with due regard given to the radar's physical capabilities.

The keyword `MTI-ATTENUATION` is used to set the transmittance of a signal as a function of the relative radial velocity of the target, the frequency of the transmitted signal and the PRF. This attenuation may be deliberate, since it is included in a 'clutter-canceller' filter designed to cut out unwanted reflections from the ground. If a passband is defined which varies across a band with width equal to the PRF, f_P, then the strength of the returned signal may be found as follows: first Suppressor computes the returned frequency, f, from the target, which is given by:

$$f = f_D - f_P \left\lfloor \frac{f_D}{f_P} \right\rfloor$$

where f_D is the modulus of the returned Doppler frequency of the echo:

$$f_D = \frac{2\,|V_R\,|}{\lambda}.$$

Here V_R is the relative speed of the target, and λ is the wavelength of the transmitted signal. Note that the factor of two arises from the fact that the signal is reflected. (This assumes that f is always positive, which further implies the target is always receding from the radar. The `MTI-ATTENUATION` table must be symmetric, i.e. the attenuation of a signal from a target approaching the radar is exactly the same as one receding at the same speed so this is not a restriction in practice.) Once f has been found the value of the attenuation at this frequency, which will always lie between zero and the PRF f_P is looked up in the `MTI-ATTENUATION` table. (This table, like the other tables dealing with losses, actually records the transmittance so that a value of unity gives no attenuation.) A typical clutter filter will eliminate returns at frequencies near the PRF and its harmonics, including zero, giving maximal transmittance near $0.5f_P$. A simple waveform for this filter could be:

$$0.5\left[1-\cos\left(2\pi f / f_P\right)\right].$$

This is designed to minimize return from ground clutter which will tend to be at relatively small velocities. The frequency spread of the clutter depends on beamwidth of the radar's mainlobe divided by the wavelength of the transmitted signal. However since the beamwidth is itself proportional to the wavelength of the transmitted signal the spread of the clutter can be minimized by using a larger antenna. In fighter aircraft, where space is at a minimum, this is rarely possible and so the PRF must be increased instead to increase the fraction of the returned signal which is useful. (The above relationship for the MTI filter would not be chosen in this case, since this wastes an equal proportion of the returned spectrum at all PRFs.)

It should be remarked here that Suppressor provides no mechanism for correcting the above relationship for those occasions when the receiver and transmitter are separate, such as with a semi-active guided missile for example. The above relationship would have to be used with the forced assumption that the relative speed of the transmitter and receiver is small.

In addition to this attenuation table a simpler mechanism exists within Suppressor to model explicit 'Doppler blind zones' or 'Doppler notches' with the keyword `SNR-DOPPLER-LIMITS`. For example the setting:

```
SNR-DOPPLER-LIMITS
  MIN-DOPPLER    50.0 (M/SEC)
  MAX-DOPPLER   450.0 (M/SEC)
END SNR-DOPPLER-LIMITS
```

would limit the radar receiver's sensitivity to those targets approaching or receding at a radial speed lying between 50 ms^{-1} and 450 ms^{-1}. This entry is actually available to all sensor receivers, not just radar receivers, although in practice its use will be largely restricted to radar receivers. For radars though it might be better to use `MTI-ATTENUATION` for this entry, since repeating bands of Doppler notches can be spread across the whole frequency range, which corresponds more closely to the operation of a real radar than the artificial setting of speed ranges. (Using SNR-DOPPLER-LIMITS allows only a single range of allowed speeds, corresponding perhaps to the first unambiguous Doppler band.) The connection between the radial speeds and the limiting frequencies is determined by the Doppler frequency. For example with a radar operating at a PRF of 750 Hz and transmitting at 3 GHz then the maximum unambiguous Doppler range of speeds is given when $f_D = f_P$ and so $V_R = \lambda f_p / 2$ or just 37.5 ms^{-1}! On the other hand, the required PRF to be able to distinguish such a wide speed range as 500 ms^{-1} with a 3 GHz transmitter would require a PRF of 10 kHz.

In addition to the entries already discussed radar receivers may have several extra options which determine how they operate when being jammed. In the DETECTION-SENSITIVITIES table the extra options available may change the example system to:

```
DETECTION-SENSITIVITIES
  SENSING-THRESHOLD                     3.0  (DB)
  RECEIVER-NOISE                     -160.0  (DB)
  INTERNAL-LOSSES                     -10.0  (DB)
  OPERATING-LOSSES                     -1.0  (DB)
  S/J-NOISE-THRESHOLD                   3.0  (DB)
  S/J-PULSE-THRESHOLD                   3.0  (DB)
  J/N-NOISE-OPERATOR-THRESHOLD        380.0  (DB)
  J/N-PULSE-OPERATOR-THRESHOLD        380.0  (DB)
END DETECTION-SENSITIVITIES
```

The values chosen above show the default settings for the four extra items. These four thresholds allow for the presence of noise and pulse jamming. The S/J-NOISE-THRESHOLD replaces SENSING-THRESHOLD as the required detection threshold for a signal once the noise jamming power is greater than the sum of the other noises present, i.e. receiver noise and clutter. If omitted its default value is in fact identical to the SENSING-THRESHOLD, here 3 dB. If pulse jamming is present and the pulse jamming signal is stronger than all the noise sources in the system, including any noise jamming, then the S/J-PULSE-THRESHOLD will replace the SENSING-THRESHOLD. Again, if omitted, the threshold will remain equal to the SENSING-THRESHOLD.

The second pair of thresholds are those which determine when the radar operator becomes aware of the presence of noise or pulse jamming. In these circumstances the operation of the radar or the tactics of the player may be altered, as described in the player-type's tactics. One option that might be taken would be to change the transmission frequency of the associated radar transmitter. The default values for these two thresholds, J/N-NOISE-OPERATOR-THRESHOLD and J/N-PULSE-OPERATOR-THRESHOLD, is 380 dB and in both cases the jamming signal must be greater by this factor than the other sources of noise, i.e. receiver noise and clutter so that the jamming can be detected.

Also of importance when a radar receiver is influenced by jamming is the RCVR-BANDWIDTH entry which sets the frequency bandwidth of the receiver. It is required for computing the effect of jammers upon the receiver, since it indicates the range of frequencies within which a jammer must be operating to influence the receiver and it allows the determination of the ratio of the received signal power to the jamming power. (Clearly 10 W of power spread across 1 kHz of the spectrum is much more effective than 1 kW spread over 1 GHz.)

In addition to the bandwidth of the receiver the possible polarization of a signal is important in modelling the effect of jamming upon a radar receiver. The power received from the jammer may be dependent upon the polarization of the transmitted disrupting signal. If so a POLARIZATION-EFFECTS table needs to be included for any receiver subject to polarized jamming signals. This has the format:

```
POLARIZATION-EFFECTS
  HORIZONTAL        0.9  (NO-UNITS)
  VERTICAL          0.8  (NO-UNITS)
  LEFT-CIRCULAR     1.1  (NO-UNITS)
  RIGHT-CIRCULAR    1.0  (NO-UNITS)
END POLARIZATION-EFFECTS
```

For example the value 0.9 indicates that 90% of any horizontally polarized jammer's power will be received by this sensor receiver. Note that regardless of the number of polarized disruptors present only one multiplicative power may be set for each of the four kinds of polarized signal. In the absence of any factor the default value is unity. These values are normally less than one, but greater values, which magnify the power received, may be set.

For any explicit jammers to be effective against a radar receiver these disruptors must be listed in a SNR-JMR-INTERACTIONS block, for example:

```
SNR-JMR-INTERACTIONS
  jadering  jukebox
END SNR-JMR-INTERACTIONS
```

would leave the radar receiver susceptible to jamming from the named explicit disruptor types 'jadering' and 'jukebox'.

Optical Sensor Receivers

Another important class of sensor receivers are optical sensor receivers. These are passive systems which use reflected light to detect their targets. As may be expected, many of the appropriate physical capabilities of these systems are very different from radar and warning receivers. An optical sensor is identified by the entry `OPTICAL` in the `SNR-CHARACTERISTICS` table, for example:

```
SNR-CHARACTERISTICS
  FREQ-DRIVEN   BOTH-ACQ/TRK   OPTICAL
END SNR-CHARACTERISTICS
```

would identify a sensor receiver as being an optical sensor receiver.

The generic items available to all types of sensor receivers may be set in the manner discussed above, but the meaning of these for the `DETECTION-SENSITIVITIES` block is different.

```
DETECTION-SENSITIVITIES
  SENSING-THRESHOLD        0.99  (NO-UNITS)
  POST-LOCKON-THRESHOLD    0.98  (NO-UNITS)
END DETECTION-SENSITIVITIES
```

The options available in the `DETECTION-SENSITIVITIES` table differ for optical sensor receivers from those for radar receivers. Only two settings are available: `SENSING-THRESHOLD` and `POST-LOCKON-THRESHOLD`. These threshold values refer to the cumulative probabilities of detection that must be achieved to successfully detect a target, the first probability `SENSING-THRESHOLD` being the threshold to initially detect a target, the second `POST-LOCKON-THRESHOLD` being that which must be maintained to continue to track the target with a tracking sensor.

To obtain the cumulative probabilities individual 'glimpse' probabilities are first computed. These refer to the probability of seeing the target at any particular instant. If the glimpse probability at some instant is P_i then the cumulative probability that the target has been detected after N such glimpses is:

$$P_N = 1 - \prod_{i=1}^{N} (1 - P_i).$$

Once P_N exceeds the sensing threshold the target is considered to have been successfully detected. If at some point the cumulative probability drops below this threshold, or below the `POST-LOCKON-THRESHOLD` when it is set for tracking sensors, the target is regarded as lost.

Before the cumulative probabilities are found we need to determine the individual glimpse probabilities. For optical sensors three different glimpse probabilities may be defined: the SEARCH-GLIMPSE-PROB during acquisition, TRACK-GLIMPSE-PROB during tracking and REACQ-GLIMPSE-PROB during attempts at reacquisition of a target recently lost sight of. All these tables have an identical format with an example of the tables being:

```
SEARCH-GLIMPSE-PROB
  DIMENSION 1 SIZE (SR)
        0.0  1.0E-8  6.3
    DIMENSION 2 CONTRAST (NO-UNITS)
        0.0     10.0
      DETECT-PROB (NO-UNITS)
            0.0
    DIMENSION 2 CONTRAST (NO-UNITS)
        0.0     10.0
      DETECT-PROB (NO-UNITS)
            1.0
END SEARCH-GLIMPSE-PROB
```

The probabilities are tabulated according to the target size (in steradians) and its inherent contrast with the environment, expressed as a real number. The above example gives no opportunity to see small items below the resolution limit of the sensor, 10^{-8} sterad, and whereas all larger items regardless of contrast are certain to be seen. Note that no object can subtend more than 2π steradians. More realistic values should be provided for real scenarios. The contrast of a target with its background is set with the INHERENT-CONTRAST item of the appropriate susceptibility block, see section 3.5. It should be noted that Suppressor assumes targets are perceived visually according to their variations in contrast, i.e. targets are not assumed to actively emit light, so the visible glow of a rocket would not be modelled by Suppressor. The influence of altitude and distance on the contrast of the target and the background are set by PATH-RADIANCE and the BACKGROUND-RADIANCE entries respectively. These tables have a common format, exemplified here by a BACKGROUND-RADIANCE table:

```
BACKGROUND-RADIANCE
  DIMENSION 1 ALT-TGT (KM)
          0.0     50.0
    DIMENSION 2 ALT-RCVR (KM)
          0.0     50.0
      DIMENSION 3 2D-DIST (KM)
          0.0     60.0
        RADIANCE (W/SR/M2)
              1.0
END BACKGROUND-RADIANCE
```

The entries are tabulated in terms of the altitude above mean sea level of the target and receiver, and the ground distance separating them. The target contrast, *C*, depends on the inherent contrast, C_i, background and path radiances, B_{BC}, and, B_p, and the transmission through the atmosphere, A_t. The relationship used by Suppressor is:

$$C = C_i \frac{A_t\, B_b}{A_t\, B_b + B_p} = \frac{C_i}{1 + B_p / A_t\, B_b}.$$

So reducing the ratio of path radiance, B_p, to the product of the transmittance, A_t, and the background radiance, B_b, increases the contrast, making the target easier to see. On the other hand a night-time mission would have a lower value of background radiance making targets much harder to see. In short a target will be easier to detect if any of the following increase: inherent contrast, C_i, transmittance, A_t, background radiance, B_b, or if the path radiance, B_p, decreases.

The atmospheric transmittance, A_t, is introduced by the `TRANSMISSION-LOSS` table. For this to be used an operating frequency must be defined for the sensor, which for optical sensor receivers is done with the `XMIT-FREQ` option, for example

```
XMIT-FREQ  6.0E3  (GHZ)
```

would specify that the optical sensor is using visible light. This frequency is then used in conjunction with a `TRANSMISSION-LOSS` table to indicate the magnitude of atmospheric losses. As with sensor transmitters this frequency may be over-ridden by the `FREQ:` entry in the SDB. (Note that in fact any frequency may be chosen so long as it uniquely identifies the portion of the `TRANSMISSION-LOSS` table to be used.)

Suppressor will not assume a target can be seen if it is hidden over the horizon or behind a topographical feature. These calculations will be done automatically once the radius of the Earth is specified. To allow for refraction at optical wavelengths an Earth radius some 20% larger than the real radius is usually taken:

```
EFFECTIVE-EARTH-RADIUS  7645.0 (KM)
```

It should be noted that the effective Earth radius must be specified independently for every sensor receiver in the model. Care should be taken that these values are all consistent. They will probably not all be equal, since the radius used varies with the wavelength of the received radiation, and Suppressor will not perform this correction automatically. If a receiver has a short enough range then little error will be caused by using a flat Earth model.

Infrared Receivers

The final class of sensor receivers that Suppressor can model are infrared sensor receivers. As with the other classes of sensor receivers an infrared sensor receiver is identified within the SNR-CHARACTERISTICS block, so that a tracking infrared receiver could have characteristics:

```
SNR-CHARACTERISTICS
  FREQ-DRIVEN   TRK   INFRA-RED
END SNR-CHARACTERISTICS
```

the keyword INFRA-RED indicating the nature of the receiver.

Infrared sensing chances are modelled in Suppressor according to the infrared cross-sectional area, σ_{IR}, of the target. This is computed by Suppressor using the equation:

$$\sigma_{IR} = I_{IR} + A\left(\frac{\alpha S}{4\pi} - B\right).$$

In this relationship I_{IR} is the infrared intensity of the target is specified, in units of W sterad^{-1}, by the IR-INTENSITY item of the susceptibility block of the target. (Note that the entry IR-RAD-TABLE is a synonym for this.) The target's projected area of the target is A measured in square metres, which is also specified in the target's susceptibility block using the OPT-CS table. Within the relationship's parentheses is a term describing the relative brightness of the target to its background. The background radiance is described by B, and set in units of W sterad^{-1}m^{-2} by the BACKGROUND-RADIANCE entry of the infrared receiver. If this table is absent the background radiance is taken to be zero. Unlike optical sensors a low background radiance increases the visibility of the target, although the entry does uses exactly the same format. The target's radiance is computed as $\alpha S / 4\pi$ where α is the target's reflectivity, i.e. its albedo, a dimensionless number specified by the target's TGT-REFLECTIVITY table. Finally S is the solar irradiance (the insolation) given in units of W sterad^{-1}m^{-2} by the SOLAR-IRRADIANCE table which is required in the definition of all infrared receivers. The overall units of the infrared cross-section are the same as those of the intrinsic intensity, W sterad^{-1}. Note that the infrared cross-section includes both intrinsic and reflection terms. So, unlike the optical cross-section where radiant light is not modelled, a source of radiant heat may be included in the model. The format of the SOLAR-IRRADIANCE table is:

```
SOLAR-IRRADIANCE
  DIMENSION 1 ALT-TGT (KM)
      0.0     50.0
    IRRADIANCE (W/M2)
         0.1
END SOLAR-IRRADIANCE
```

so that the insolation is defined as a function of the target's altitude.

An infrared receiver's `DETECTION-SENSITIVITIES` may have three options set: `SENSING-THRESHOLD`, `POST-LOCKON-THRESHOLD` and `RECEIVER-NOISE`. So, for example:

```
DETECTION-SENSITIVITIES
  SENSING-THRESHOLD          3.0  (NO-UNITS)
  POST-LOCKON-THRESHOLD      3.0  (NO-UNITS)
  RECEIVER-NOISE         6.65E-8  (W/M2)
END DETECTION-SENSITIVITIES
```

The value of these options takes yet another form with this class of receiver. The `RECEIVER-NOISE` entry is the noise equivalent irradiance of the receiver in units of W m^{-2}. In computing a sensing chance the received infrared signal must be compared with this intrinsic noise figure. If the resulting ratio is greater than the `SENSING-THRESHOLD`, or (when tracking) the `POST-LOCKON-THRESHOLD` the detection is successful. Since the separation R of the target and the receiver is known from the geometry of the situation then the energy flux stemming from the target at the receiver is just $4\pi\, \sigma_{IR} \,/\, R^2$, which can then be used in the sensing threshold comparison.

If the receiver is close enough to the target that only a portion of the target's surface area may be contained in the receiver's field of view then the effective size and thus the radiance of the target should be reduced appropriately. Suppressor accomplishes this by ensuring that the angle subtended by the target is never larger than the `PIXEL-FIELD-OF-VIEW` item which may be set for the receiver. For example, the entry:

```
PIXEL-FIELD-OF-VIEW        6.28 (SR)
```

would ensure that the computed solid angle of the target never exceeds 2π steradians, which of course should always be true.

3.4.5 Disruptors

Disruptors are used to interfere with other players' abilities to function. ECM jammers and simple smoke flares would both be examples of disruptors. Unlike weapons, which are used to destroy the constituent elements of players, disruptors seek to incapacitate the basic physical systems making up other players. Weapons play a role in lethal engagement, whereas disruptors are the essential systems for nonlethal engagements.

Suppressor models disruptors in two distinct ways: either purely probabilistically as so-called implicit disruptors, or in detail as explicit disruptors. Only those disruptors which operate in the radio frequency (RF) range may be treated explicitly. RF disruptors, e.g. jammers, may thus be modelled either implicitly or explicitly, whereas the effects of chaff and flares may only be modelled implicitly. The nature of a disruptor may often be deduced from the system's definition in the player-type template. Explicit disruptors will always have the following form:

```
DISRUPTOR  1050 jammer  CAPABILITY    jammer_data
```

which uses no expendable countermeasure stocks. Implicitly modelled disruptors, such as the following flare system *usually* do use expendable countermeasures:

```
DISRUPTOR    1051 flare_launcher
  CAPABILITY     flare_launcher_data
  CM-EXPENDABLE flare DISCRETE 40 (NO-UNITS)
```

In the CAPABILITY block of the disruptor the DISRUPTOR-CHARACTERISTICS definitions must always be included. These spell out whether the disruptor is implicit or explicit in nature using the keywords IMPLICIT-DETAIL or EXPLICIT-DETAIL. Another, redundant, option specifies whether the disruptor acts passively or actively; all EXPLICIT-DETAIL disruptors are defined as ACTIVE, and all IMPLICIT-DETAIL as being PASSIVE.

A simple implicit disruptor, such as '1051 flare_launcher', will have the following characteristics:

```
DISRUPTOR-CHARACTERISTICS
  IMPLICIT-DETAIL  PASSIVE
END DISRUPTOR-CHARACTERISTICS
```

Implicit disruptors can only impact the performance of weapon and sensing systems probabilistically. Used against weapons they can influence the WPN-PK and WPN-TIME-DELAYS entries of weapons used to attack locations where an implicit disruptor is available as a defensive measure. Against sensor systems the EFF-BURST-CM-PROBEFF-BURST-CM-PROB item can be used to influence the success of a modelled sensing chance. Note further that all disruptors, both implicitly and explicitly modelled ones, only influence weapons when they can be used to disrupt the tracking sensor receiver linked to the weapon. All disruptors using chaff or flares are implicit disruptors, as are ECM disruptors such as jammers whose workings are not required to be modelled in detail.

When ECM disruptors are indeed modelled in detail they must be included as explicit disruptors in the model. In these cases the amount of energy emitted by the jammer that is interfering with the targeted sensor and communication receivers will be explicitly calculated by Suppressor. The DISRUPTOR-CHARACTERISTICS block becomes more complex for explicit jammers since extra information is required. First of all the mode of operation must be selected, the choice being between NOISE and PULSE modes. In NOISE mode the disruptor will model jammer systems whose goal is to swamp the enemy's signals by creating excessive noise. A receiver operating on the targeted frequency will receive an interference signal consisting of receiver noise, clutter noise plus the noise jamming signal, against all of which the true signal will have to compete. Noise jamming is effective against both radar and communication receivers. (It is worth mentioning at this point that explicit disruptors target emitters, i.e. sensor transmitters or communication transmitters. However their emissions will interfere with signals detected either by sensor receivers or communication receivers. The theory is that if there is a transmitter emitting a signal there must be a receiver listening in somewhere.)

An example of a characteristic block for an explicit disruptor using noise jamming is:

```
$ A vertically polarized explicit noise jammer
DISRUPTOR-CHARACTERISTICS
  EXPLICIT-DETAIL  ACTIVE
  NOISE
  VERT-POL
END DISRUPTOR-CHARACTERISTICS
```

Which, apart from specifying that the disruptor is an explicit noise mode jammer, also specifies that it transmits vertically polarized emissions. If the targeted receiver has a different polarization to that of the jammer the effectiveness of the jamming will be reduced. Four distinct polarizations may be chosen for either noise or pulse jammers:
- VERT-POL for vertically polarized signals,
- HORIZ-POL for horizontal polarization,
- LEFT-CIR-POL for left circular polarization, and
- RIGHT-CIR-POL for signals with a right circular polarization.

If this keyword is omitted then the disruptor will emit unpolarized signals.

The second mode of operation, PULSE mode, models jammers which send out a phoney signal. Since pulse jamming concentrates power into narrow pulses it requires less powerful transmitters than does noise jamming. It is, however, only effective against radar receivers. When PULSE jamming is selected an extra DISRUPTOR-CHARACTERISTICS item becomes mandatory, that selecting the disruptor's power distribution. This is either CONST-PWR/SPOT or AVG-PWR/SPOT. Each active jammer focuses energy onto one or more 'spots' in the frequency band. The number of spots that may be focused simultaneously by the jammer is set with the MAX-NO-SPOTS capability of the disruptor, for example:

```
MAX-NO-SPOTS 2 (NO-UNITS)
```

If the jammer is specified as having a CONST-PWR/SPOT then each spot will have a fixed power focused upon it, with the data item MAX-POWER-OUT setting the value of this power, for example:

```
MAX-POWER-OUT 7800.0 (WATTS)
```

All NOISE jammers function in this way but PULSE jammers may also allocate their available energy differently using the AVG-PWR/SPOT option. Here MAX-POWER-OUT represents the jammer's total power and each spot gets an equal share of the power available. An example characteristics block for a pulse jammer could be:

```
$ An unpolarized explicit pulse jammer
DISRUPTOR-CHARACTERISTICS
  EXPLICIT-DETAIL  ACTIVE
  PULSE
  AVG-PWR/SPOT
END DISRUPTOR-CHARACTERISTICS
```

Modelling active explicit disruptors requires that several physical capabilities be defined. Firstly, their operational frequencies must be set using DISRUPTOR-FREQ-LIMITS. The lower and upper frequency bounds are set using LOWER-FREQ-LIMIT and UPPER-FREQ-LIMIT. If the jammer is to be used reactively under the control of tactics defined for nonlethal engagements then the SPOT-SIZE-BANDWIDTH should also be set. This determines the size of the frequency spots used against target systems. Finally, for jammers which operate in PULSE mode the SUBCARRIER-BANDWIDTH should be given too, as sub-carriers are used collectively to cover a wide spot. An example setting for a reactive noise jamming system is thus:

```
DISRUPTOR-FREQ-LIMITS
  LOWER-FREQ-LIMIT       1.11  (GHZ)
  UPPER-FREQ-LIMIT       4.23  (GHZ)
  SPOT-SIZE-BANDWIDTH   70.00  (MHZ)
END DISRUPTOR-FREQ-LIMITS
```

A further requirement for active explicit disruptors is that the maximum physical range of the jammer is set, for example:

```
MAX-RNG            185.0 (KM)
```

Jamming is effective when more power arrives at a receiver from the jammer than from the signal, in other words when the signal to jamming ratio (S/J) is small. These powers represent the spectral power density of the jammer and of the signal. (Spectral power density is the total power of a transmitted signal divided by the bandwidth of the transmission.) Thus the comparison of the signal and jamming strengths uses both the bandwidth of the disruptor, defined by SPOT-SIZE-BANDWIDTH and the bandwidth of the transmitted signal it is competing with, defined by the RCVR-

BANDWIDTH entry of the radar or radio receiver. If the jamming signal is strong enough it will dilute the receiver's signal to noise (S/N) ratio sufficiently to negate an otherwise good sensing chance, since the S/N ratio would now be below the relevant sensing threshold as defined in the DETECTION-SENSITIVITIES block of the receiver.

The determination of just how much jamming power is being received by a sensor or communication receiver depends on the source of the signal, i.e. whether the jamming signal is from pulse or noise jammers or a mixture of both. In the case of noise jamming the receiver sums all the power received at the appropriate frequencies from all the noise jammers that are within range. The signals from the jammers are assumed to be emanating from the jammers' mainbeams unless any jammer is constrained to point away from the receiver by a POINT IT instruction in the SDB; in this case the jammer's sidelobes will be used. The normal constraints on the angle of the receiver's antenna are also applied, (viz. POINT IT and slew limits when appropriate), to see if the signal is detected by the receiver's mainbeam or a sidelobe. This summed signal is the total noise jamming signal interfering with the signal. Extra noise due to background clutter and receiver noise itself may also be present. With pulse jamming only the signal from a single PULSE mode jammer is effective, and this is the strongest such signal arriving at the receiver. If this signal is weaker than any noise signal that may also be present the pulsed jamming signal is disregarded, otherwise the jamming signal is that from the pulsed jammer.

It should be now clear that Suppressor cannot model deceptive jamming which fools a player to perceive nonexistent targets. The best Suppressor can do is to prevent the detection of the real target. That is to say even though a jamming signal may enter a sensor receiver's sidelobes, and even if this signal is much larger than that of a genuine target in the mainbeam, it still cannot cause Suppressor to infer falsely a target azimuth but merely prevent the detection of the genuine target.

Just as sensor and communication transmitters can try to avoid perceived jamming by changing their frequency, disruptor's may themselves react to counter this tactic. If the item TIME-REACT-FREQ-CHANGE is defined then after this interval has expired the disruptor may try to jam the transmitter's new frequency. If this time is omitted or set to be negative the disruptor will not be able to change its frequency.

Several other options related to the modelling of RF transmissions and transmitters can be included in a disruptor's CAPABILITY block. All these have been discussed in some detail in the sections of this guide describing sensor receivers and sensor transmitters. These are: VERTICAL-OFFSET which is useful for ground based explicit jammers, ANTENNA-PATTERN for defining the disruptor's beam pattern, in addition, LAMBDA-PATTERN and SINE-PATTERN may also be used for this purpose. The command INTERNAL-LOSS defines the efficiency of the disruptor, whilst TRANSMISSION-LOSS allows the computation of atmospheric signal losses; and EFFECTIVE-EARTH-RADIUS is used for computing the horizon distance from the disruptor. In addition jamming signals can be masked by terrain in just the same way as signals to and from sensors, and indeed, signals from communication systems, to which we will turn next.

3.4.6 Communication Systems

The generic function of talking is modelled in Suppressor using communication receivers and communication transmitters. These systems may function by using either landlines (implicit nets) or radio broadcasts (explicit nets). Broadcast communication systems, i.e. explicit nets, may be disrupted by jamming or masked by terrain, and they may also be sensed by other players' warning receivers. Implicit nets are immune to these difficulties and dangers but are much less flexible and cannot be used with moving players. Communication networks are a vital component of all C^3I structures in any Suppressor model. If a player is to be able to communicate fully it requires both communication receivers and transmitters. With transmitters alone it can pass out instructions and information; with a receiver alone it may listen to messages.

For implicit nets the capability blocks of the communication receivers and transmitters are entirely redundant. Landlines cannot be jammed or overheard by warning receivers, and they do not need to use antenna and so do not require the setting of operational frequencies and bandwidths. However communication receivers and transmitters that are defined for use on explicit nets may also be employed on implicit nets. The extraneous information simply being ignored. Suppressor identifies which nets are which using the `NET TYPE` instruction in the SDB file which names each net. The nets' names themselves are declared in the UAN as being the names of entities which are either `IMPLICIT-NETS` or `EXPLICIT-NETS`. All the following `CAPABILITY` information is therefore only needed for communication systems that are to be used on explicit nets.

3.4.6.1 *Communication Transmitters*

A communication transmitter has many similarities in its definition to that of a sensor transmitter. An example CAPABILITY block for a radio transmitter is:

```
CAPABILITY  radio_tx_data
  VERTICAL-OFFSET 300.0 (M)
  XMTR-BANDWIDTH    2.0    (MHZ)
  XMTR-POWER       50.0E3 (WATTS)
$ A simple omni-directional antenna
  ANTENNA-PATTERN
    DIMENSION 1 AZ (DEG)      0.0  180.0
      DIMENSION 2 EL (DEG) -90.0   90.0
        GAIN (DB)  30.0
  END ANTENNA-PATTERN
  INTERCEPT-INTERACT
    finder_rx
  END INTERCEPT-INTERACT
END CAPABILITY
```

The VERTICAL-OFFSET, ANTENNA-PATTERN and INTERCEPT-INTERACT, which lists the potential eavesdropping warning receivers, have exactly the same function as for sensor transmitters. The standard antenna patterns which sensor transmitters may use, such as LAMBDA-PATTERN, are not valid options for communication receivers, however. For explicit nets both the broadcast power, XMTR-POWER, and the bandwidth, XMTR-BANDWIDTH, must be specified.

The operational frequency of the communication system is not set within the TDB but by the FREQ: entry in the SDB pertaining to the system. Care should be taken that different player's communication systems are in fact operating on the same frequency.

3.4.6.2 *Communication Receivers*

Communication receivers on explicit nets require the use of antennae, and so, like sensor systems the options describing the antennae are shared with the transmitters. These are ANTENNA-PATTERN and VERTICAL-OFFSET. Furthermore, like sensor-receivers, the option EFFECTIVE-EARTH-RADIUS may be used to set the horizon distance and the TRANSMISSION-LOSS to describe atmospheric losses of the signal.

Just as the receiver noise is required for radar sensor receivers, so that the operational range may be determined, so is it also required for communication receivers. Here the noise is set by the keyword RCVR-NOISE:

```
RCVR-NOISE 1.0E-10 (WATTS)
```

An incoming signal will be recognizable if its signal strength is greater than the total receiver noise and any jamming noise by a factor specified by the `RECOGNITION-THRESH` item. For example if

```
RECOGNITION-THRESH 3.0 (DB)
```

is set then the signal must be at least twice as strong as all noise signals arriving at the receiver to be comprehensible. The quantities compared in determining the signal to noise ratio are the spectral power densities, so the bandwidth of the receiver is required in computing this figure. This is specified with the `RCVR-BANDWIDTH` item:

```
RCVR-BANDWIDTH 2.0 (MHZ)
```

The jammers that may disrupt the receiver, (which must be active explicit noise jammers), are identified with the `COMM-JMR-INTERACTIONS` block:

```
COMM-JMR-INTERACTIONS
  jukebox  jadering
END COMM-JMR-INTERACTIONS
```

(Note that there is no method allowing implicitly modelled jammers to interfere with communication devices.) The final data item is the jamming signal to noise ratio threshold at which the communication receiver's operator will become aware of the jamming and attempt to take countermeasures (such as switching to an alternative frequency listed under the `ALT-FREQ:` entry in the SDB). This is set using the `J/N-COMM-OPERATOR-THRESHOLD` keyword. If omitted it has a default value of 380 dB, so that the entry:

```
J/N-COMM-OPERATOR-THRESHOLD 380.0 (DB)
```

would not in fact change the default value.

As with sensor receivers the receiver's susceptibility to jamming with polarized emissions can be set with the entry `POLARIZATION-EFFECTS`. This has the same syntax and function as for radar sensor receivers.

3.5 The Susceptibility Block

The role of the SUSCEPTIBILITY block is to define, for each element of a player-type, just how likely this element is to be seen by a particular class of sensor receiver. If the SUSCEPTIBILITY entries are absent then the element will be invisible to all sensors in the scenario. The name of each element's SUSCEPTIBILITY block, which may be shared with other elements, is identified in the player-type template in the following manner:

```
PLAYER-STRUCTURE target
  LOCATION  1
    ELEMENT 10 bridge         CONTINUOUS 1.0
      SUSCEPTIBILITY bridge_signature
END PLAYER-STRUCTURE
```

The SUSCEPTIBILITY item must explicitly name those sensor systems which may detect the element, and if no systems are named the element will be invisible. These sensors are identified with the SNR-ELE-INTERACTIONS block, for example:

```
SUSCEPTIBILITY bridge_signature
  SNR-ELE-INTERACTIONS
    bomber_radar_rx
    ground_attack_helicopter_optical_rx
    ground_attack_helicopter_infrared_rx
    fighter-bomber_radar_rx
  END SNR-ELE-INTERACTIONS
  <omitted radiation cross-section tables>
END SUSCEPTIBILITY
```

The 'bridge' element of the player-type 'target' can be sensed by several systems, viz. radar sensor receivers carried by bombers and a fighter-bombers, and optical and infrared detectors carried by helicopters. Since a bridge (presumably) emits no radio frequency signals no warning receiver is able to detect it. Despite this the bridge is visible in all the wavebands modelled by Suppressor, and so detection cross-sections must be defined for it for in the optical, infrared and radio wavelengths.

The bridge's radar cross-section is defined by the RCS-TABLE entry, which specifies its radar cross-section. In its simplest form it appears as:

```
RCS-TABLE
  DIMENSION 1 AZ (DEG)      0.0  180.0
    DIMENSION 2 EL (DEG) -90.0   90.0
       RCS (M2)              1.0
END RCS-TABLE
```

The azimuthal and angular dependence of the radar cross-section must always be given. The azimuthal dependence is assumed to be symmetric around the zero azimuth if only positive azimuths are specified. The reference azimuth is due east or the target's heading. If, like the bridge, the target is motionless then the reference azimuth is assumed to point due east, although this can be over-ridden with the HDG: entry of the LOC: command in the SDB file. Asymmetric signatures may be defined by specifying the full range of azimuthal angles from -180° to 180°. Elevation ranges must always be given for the full range of -90° to 90°, symmetry never being assumed. The order of the two entries may be reversed with elevation given before azimuth.

The above radar cross section is 'grey', that is to say it has no frequency dependence. Both frequency dependence and the dependence upon the polarization of the radar signal may be specified by two further lists in the RCS-TABLE: FREQ and POL. These must precede the angular dependence lists but apart from this constraint they may be in any order. So for example a surface which is only highly reflective to horizontally polarized radio waves with frequencies near 10 GHz could have the following entry:

```
RCS-TABLE
  DIMENSIONS 1 POL        HORIZ-POL  DEFAULT
    DIMENSION 2 FREQ (GHZ) 0.0   9.0   11.0 100.0
      DIMENSION 3 AZ (DEG)      0.0  180.0
        DIMENSION 4 EL (DEG) -90.0   90.0
          RCS (M2)               0.0
      DIMENSION 3 AZ (DEG)      0.0  180.0
        DIMENSION 4 EL (DEG) -90.0   90.0
          RCS (M2)               1.0
      DIMENSION 3 AZ (DEG)      0.0  180.0
        DIMENSION 4 EL (DEG) -90.0   90.0
          RCS (M2)               0.0
    DIMENSION 2 FREQ (GHZ)      0.0  100.0
      DIMENSION 3 AZ (DEG)      0.0  180.0
        DIMENSION 4 EL (DEG) -90.0   90.0
          RCS (M2)               0.0
END RCS-TABLE
```

The DEFAULT entry specifies the reflectivity of the element to unpolarized radio waves, and must always be present in the list of polarizations, POL.

Optical sensors use the contrast between a target and its background to perceive the target. The greater the contrast and the larger the angular size of the target the easier the target is to see. The contrast is defined by the value of the INHERENT-CONTRAST entry, for example:

```
INHERENT-CONTRAST  1.0 (NO-UNITS)
```

The size of the target is specified as a function of its geometrical aspect, in terms of the target's projected area in square metres. So in the unlikely event of a perfectly spherical bridge with a projected surface area of 10 square metres its optical cross-section would be set by the OPT-CS table as:

```
OPT-CS
  DIMENSION 1 AZ (DEG)      0.0  180.0
    DIMENSION 2 EL (DEG) -90.0  90.0
      OCS (M2)              10.0
END OPT-CS
```

Cross-sections that are asymmetrical in azimuth may be defined by giving an azimuthal range running from -180° to 180°. Unlike radio frequencies colour dependence and polarization effects are not modelled for optical sensors. (Polarized sunglasses are not standard equipment for Suppressor pilots.)

Infrared detectors require the setting of three entries. First of all the OPT-CS table is again useful, this time to set the physical area of the target. Subsequent to this a TGT-REFLECTIVITY table gives the target's albedo, i.e the fraction of the ambient radiation that is reflected by the target. This may vary from one to another of several named infrared frequency bands. The FREQUENCY-BAND instruction defines the names of the bands to be used, for example:

```
FREQUENCY-BAND K-BAND J-BAND
```

The band names are user defined, and do not have to be associated with numerical frequency values. However, the names chosen must have been declared in the IR-BAND entry of the UAN database. If no frequency bands are defined the DEFAULT band will be chosen in the TGT-REFLECTIVITY and the IR-INTENSITY table items.

A simple TGT-REFLECTIVITY table using the DEFAULT band is shown below:

```
TGT-REFLECTIVITY
  DIMENSION 1 IR-BAND  DEFAULT
    REFLECTANCE (NO-UNITS)
       1.0
END TGT-REFLECTIVITY
```

The above example would give the target a perfect albedo, i.e. it reflects all the energy that reaches it in all of the infrared region.

Unlike optical sensors, which only detect reflected light, Suppressor can model the intrinsic infrared luminosity of the target. This is set with the IR-INTENSITY table. If the target's intrinsic infrared intensity is independent of frequency it may appear as:

```
IR-INTENSITY
  DIMENSION 1 AZ (DEG)        0.0  180.0
    DIMENSION 2 EL (DEG)    -90.0   90.0
       IR-RAD (WATTS/STERADIAN) 475.0
END IR-INTENSITY
```

The azimuthal and elevation lists may be inverted and an azimuthally asymmetric signature defined by varying the azimuth from -180° to 180°. A third optional list argument, specifying the infrared waveband, may precede the angular lists using the description IR-BAND. Also the keyword IR-RAD-TABLE is synonymous with IR-INTENSITY. Thus, the following table is exactly equivalent to the above:

```
IR-RAD-TABLE
  DIMENSION 1 IR-BAND DEFAULT
    DIMENSION 2 EL (DEG)      -90.0   90.0
      DIMENSION 3 AZ (DEG)  -180.0  180.0
        IR-RAD (WATTS/STERADIAN)  475.0
END IR-RAD-TABLE
```

3.6 The Tactic Block

The tactics employed by players within Suppressor are entirely determined by the contents of the TACTIC block, one of which is associated with each player-type's template:

```
$ Portion of a PLAYER-STRUCTURE for an AEW craft
PLAYER-STRUCTURE aew_player
  TACTIC   aew_tactics
  LOCATION 1
    ELEMENT      10 aew_craft   DISCRETE 1
      <Definitions for AEW player>
END PLAYER-STRUCTURE
```

The above example shows the format of the template's declaration of a TACTIC block, here 'aew_tactics' of the 'aew_player'. A single TACTIC block occurs for each player-type, and its contents function as a program which controls the actions of all the elements of each of the players of this type.

The program encoded by the player-type's tactics will be enacted by the thinkers of each player of this type, whose capabilities effectively define the 'processing power' of each player. Since all these thinkers share the same tactics, and hence the same program, care must be taken when multi-element player-types are defined. All the thinkers of a single player, even when distributed over many physical locations, share all the player's perceptions. These thinkers are effectively able to communicate perfectly and instantaneously with each other at all times.

Each player-type template can represent several players in the scenario. For example a player-type representing an F16 fighter may be used to define the properties of several F16 fighters appearing on more than one side of the scenario. Since a single `TACTIC` block is used for each player-type in the TDB this would imply that the tactics of all these players would be the same. This could well be a problem, since clearly the role of players on different sides of a conflict are unlikely to be identical. If they differ significantly different player-type templates will needed for different roles, for example, F16A for one role and F16B for another. If their differences are small it may be preferable to design a flexible `TACTIC` block to handle all the players. To assist in this, the `TACTIC` block has several logical switches to identify each player's current status; furthermore each player may be initialized with specific initial conditions. This information can be used to identify each player and differentiate their functions, for example whether a player is to be used in an offensive or defensive role.

Each player's tactics are built up out smaller blocks which have a specific function. However, unlike the player's capabilities where there is a one-to-one relationship between systems and the generic functions, the tactical blocks interact with these functions and depend upon different events in a complex manner. Suppose, for example, that a player is destroyed during the scenario, then as a consequence of this many different events requiring mental processing will occur. Perceptions will need to be deleted, engagements stopped, pending messages cancelled, movement plans curtailed, intelligence reporting re-routed and so on. For our discussion we will therefore group the tactical capabilities under the following three headings:

- Co-ordination, tactics dealing with command, control and intelligence issues,
- Movement, tactics instructing players when and where to move, and
- Resource Allocation, how and when a player should employ its resources, i.e. its physical capabilities apart from movement.

The last two of the above groupings can be identified with specific tactics within each player's `TACTIC` block. Thus, movement is associated with the `MOVE-PLANS` tactics, and the allocation of a player's resources with the `RESOURCE-ALLOCATION` block. This latter block is complex since it deals with most of the tactical decisions taken by players in the 'reacting' phase of thinking. (This is the mental activity which determines what actions a player will take when faced with new or changed perceptions.) In fact `RESOURCE-ALLOCATION` is complex enough to be itself subdivided into six distinct tactics, each controlling a particular aspect of the behaviour of the player. These areas, along with the tactics of reactive movement, were in fact introduced earlier in this guide whilst defining the capabilities of thinkers. They are:

- lethal assignment, making assignments to subordinates to attack other players, (identifier LETHAL-ASSIGNMENT);
- lethal engagement, using weapons to attack other players or systems, (LETHAL-ENGAGE)
- non-lethal engagement, using disruptors to reactively jam communication or sensor receivers, (NONLETHAL-ENGAGEMENT);
- emission control, turning sensor systems on or off, (EMCON);
- lethal mode of control, defining the authority needed for subordinates to lethally engagement targets, (GUNS-FREE/GUNS-TIGHT);
- launching movement, launching subordinate players into motion, (LAUNCH-START).

There is a seventh area of reactive thinking, namely reactive movement, controlled by the MOVE-PLANS tactics. Both RESOURCE-ALLOCATION and MOVE-PLANS have some common features, with both being controlled by user defined procedural scripts which describe what each player is to do in various situations. This is accomplished with the help of a large number of logical criteria which each thinker uses in analysing the current situation. The following section describes all these criteria, along with where in each block of user defined tactics they may be used.

3.6.1 Tactical Criteria

The format of each criterion depends on the nature of the variable being tested. If the variable takes only a few possible values then many criteria are formed with the IS and IS NOT relationship. An example is the BEEN-ASSIGNED criterion, which tests if the current target has been assigned to the current player by its commander, and will therefore be either true or false. All the possible formats for this criterion are:

BEEN-ASSIGNED IS YES	BEEN-ASSIGNED IS NOT NO
BEEN-ASSIGNED IS NO	BEEN-ASSIGNED IS NOT YES

The two entries in each row are equivalent, both yielding the same result in all circumstances.

The above syntax can be extended quite easily to cases when three or more values are allowed. An example of this is the JAMMER-STATUS criterion which tests the operational state of a disruptor; these states are JMR-OFF when the jammer is off; JMR-ON when it is on and JMR-NON/OP when it is not operational. Six possible conditions can be tested, although in this case no two conditions are exactly equivalent.

JAMMER-STATUS IS JMR-OFF	JAMMER-STATUS IS NOT JMR-OFF
JAMMER-STATUS IS JMR-ON	JAMMER-STATUS IS NOT JMR-ON
JAMMER-STATUS IS JMR-NON/OP	JAMMER-STATUS IS NOT JMR-NON/OP

The criterion `3D-TGT-LOC`, which is used to test if a target is within a named zone, uses the terms `WITHIN` and `OUTSIDE` to indicate the nature of the test. So the criterion `3D-TGT-LOC WITHIN sam_command_zone` tests that a target is within the zone 'sam_command_zone', whereas `3D-TGT-LOC OUTSIDE my_zone` would see if a target were outside the zone 'my_zone'.

A few criteria need no further qualification apart from their title and are simply either true or false; these are `BELIEVED-ALIVE`, `BELIEVED-DEAD`, and `INTERCEPT-RESULTS-KNOWN`. So, for example, `BELIEVED-DEAD` is true when the target referred to is believed dead by the player considering the criterion.

Many criteria test quantities which take numerical values, which may be either discrete, and so integer valued, or continuous and therefore real valued. Within `RESOURCE-ALLOCATION` procedures criteria taking integer values are tested with the help of the keywords `AT-LEAST` and `NO-MORE-THAN`, e.g. `TOTAL-TARGETS AT-LEAST 4 (TGTS)` would be true when a player perceives four or more targets. The opposite condition to this would be `TOTAL-TARGETS NO-MORE-THAN 3 (TGTS)`, which would be false when there are four or more targets. For real valued items, such as measures of speed and distance, the tests are formed with the greater than '>' and less than '<' symbols. An example is `3D-DIST > 30.0 (KM)` which is true when a target is more than 30 km away.

The `MOVE-PLANS` tactics treat all items which take numerical values as being real valued, even when they are in fact discrete quantities. So, confusingly, the format of a few criteria are different. For instance the test to see if four or more targets can be seen is `TOTAL-TARGETS > 3.0 (TGTS)`, i.e. a test of whether more than three targets can be seen. The logical converse can be specified with `TOTAL-TARGETS < 4.0 (TGTS)`. Notice that the numerical values of these tests are different than for the corresponding `RESOURCE-ALLOCATION` tests. To avoid problems with rounding errors it may be safer, albeit slightly paradoxical, to use fractional values in the tests, for example, to encode a test of no more than four targets as `TOTAL-TARGETS < 3.5 (TGTS)`.

Several criteria can be qualified with a phrase following a keyword `RE:` to more closely define the meaning of the test. For example, the number of targets of a particular player-type can be specified; so `TOTAL-TARGETS > 1.5 (TGTS) RE: F16 TGT-TYPE` would be a `MOVE-PLANS` criterion requiring at least two targets of the F16 player-type be perceived.

The MOVE-PLANS tactics can use dummy arguments in their definition which are supplied with real values by the SDB. This is a very useful feature since it allows different players of the same player-type to behave differently. The actual value of these arguments can be tested in the player's MOVE-PLANS to modify the player's behaviour appropriately. So, for example, if a MOVE-PLAN tactic were defined with the argument name 'All_threats' whilst the plan was initialized at run time with the SDB value 'air_threats' then the test:

```
WHEN All_threats IS air_threats
```

would be true.

The criteria themselves are listed below, along with an example and a description for each. Not all criteria can be used in all possible tactical decisions. Which criteria can be used where is indicated by the following abbreviations:

Abbreviation	Tactical Blocks
MOV	MOVE-PLANS
ENG	LETHAL-ENGAGE
ASG	LETHAL-ASSIGNMENT
JAM	JAMMER
EMCON	EMCON
LAUNCH	LAUNCH-START
GUNS	GUNS-FREE/GUNS-TIGHT

The tactical types listed in the above table correspond to the six RESOURCE-ALLOCATION tactical classes described earlier along with the MOVE-PLANS tactics of reactive movement.

Where differences exist between the criteria in the MOVE-PLANS and RESOURCE-ALLOCATION blocks these are indicated in the text.

- ACTIVE-ATTACK-PRIORITY
  Tactics: ASG, ENG, JAM.

```
ACTIVE-ATTACK-PRIORITY IS YES
```

ACTIVE-ATTACK-PRIORITY tests if a target element is a member of the current ATK-PRIORITIES list set by the player's MOVE-PLANS. When used with the non-lethal engagement resource allocation procedures the target is a system, not an element, and so will be true when the targeted systems are part of an element in the attack priority list. This criterion is important since it is a mechanism for giving two players of the same player-type different tactical behaviour. (Different players of the same type can have different current attack priorities specified within the SDB data-file).

- AVAILABLE-RESOURCE
  Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH

```
AVAILABLE-RESOURCE AT-LEAST 6 (ROUNDS)
  RE: missile ORDNANCE
```

Tactic: MOV

```
AVAILABLE-RESOURCE > 5.9 (ROUNDS)
  RE: ALL ORDNANCE
```

AVAILABLE-RESOURCE is used to check the number of rounds of ammunition remaining. It may be qualified to specify a particular kind of ordnance or be used to consider all available ammunition. The meaning depends on the procedure being executed; for lethal engagement tactics (ENG) it specifies the amount of ammunition available to the weapon named in the procedure. For tactics controlling subordinates, (ASG, LAUNCH and GUNS) it refers to all weapons belonging to the relevant subordinates. For all other tactics (JAM, EMCON and MOV) it refers to all the weapons belonging to the player evaluating the procedure.

- BEEN-ASSIGNED
  Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH, MOV.

```
BEEN-ASSIGNED IS YES
```

BEEN-ASSIGNED tests if the current target is assigned to the player (for ASG, ENG, JAM and MOV tactics) and if any target is assigned to the player, (for GUNS, EMCON and LAUNCH tactics).

- BELIEVED-ALIVE
  Tactics: ASG, ENG, JAM.

```
BELIEVED-ALIVE
```

BELIEVED-ALIVE is true if the player believes the target is alive.

- BELIEVED-DEAD
  Tactics: ASG, ENG, JAM.

```
BELIEVED-DEAD
```

BELIEVED-DEAD is true if the player believes the target is dead.

- COMM-FREQ
  Tactics: JAM.

```
COMM-FREQ > 5.0 (MHZ)
```

COMM-FREQ tests the transmission frequency of the current target communication transmitter.

- CURRENT-ASGS
  Tactics: ASG, GUNS, LAUNCH.

```
CURRENT-ASGS NO-MORE-THAN 5 (TGTS)
```

CURRENT-ASGS checks the number of assignments made to the current subordinate (ASG) or all subordinates (GUNS, LAUNCH).

- CURRENT-ENG'S
  Tactics: ENG, ASG, EMCON, JAM, GUNS, LAUNCH.

```
CURRENT-ENG'S AT-LEAST 4 (TGTS)
```

CURRENT-ENG'S checks the number of current engagements for the current weapon system, (ENG); the current subordinate or subordinates (ASG, LAUNCH, GUNS); or for the thinker's player (EMCON, JAMMER).

- CURRENT-SPOTS
  Tactic: JAM.

```
CURRENT-SPOTS NO-MORE-THAN 2 (TGTS)
```

CURRENT-SPOTS checks the number of spots being used by the current jammer.

- CURRENTLY-JAMMED-FOR
  Tactics: ENG, ASG, EMCON, GUNS, LAUNCH.

```
CURRENTLY-JAMMED-FOR < 8.0 (SEC)
```

CURRENTLY-JAMMED-FOR checks the amount of time a resource has been jammed. If the resource is not jammed it is set a time of -1.0 s. The resource in question is the tracking sensor linked to the current weapon which has been jammed for the longest period, (ENG); or the sensor which has being jammmed for the longest time that belongs to a relevant subordinate, (ASG, LAUNCH, GUNS); or the current sensor, (EMCON).

- ELEV-ANGLE-TO-TGT
  Tactics: ASG, ENG, JAM, MOV.

```
ELEV-ANGLE-TO-TGT > 35.0 (DEG)
```

ELEV-ANGLE-TO-TGT tests the elevation angle subtended by the target, referred to as the pitch angle of the resource. If the resource is moving the pitch angle is the angle of climb or descent. For non-moving resources the pitch angle is always zero.

- EMCON-CONTROL-MODE
  Tactics: ENG, ASG, EMCON, JAM, GUNS, LAUNCH.

```
EMCON-CONTROL-MODE IS SELF
```

EMCON-CONTROL-MODE should allow a player to check if it has the authority to turn on and off its sensors without receiving authority from a commander. It is, however, currently unimplemented in version 5.3 of Suppressor. The initial status of this mode is set in the SDB using the MODES-OF-CONTROL: item EMCON.

- ENG-CONTROL-MODE
  Tactics: ENG, ASG, EMCON, JAM, GUNS, LAUNCH.

```
ENG-CONTROL-MODE IS SELF
```

ENG-CONTROL-MODE allows a player to check if it has the authority to engage a target without receiving authority from a commander. The status of this mode is initially set in the SDB using the MODES-OF-CONTROL: item ENGAGE, it can then subsequently be changed with the lethal mode of control tactics, GUNS-FREE and GUNS-TIGHT.

- FIRED-BEFORE
  Tactics: ASG, ENG, JAM.

  ```
  FIRED-BEFORE IS YES
  ```

  FIRED-BEFORE checks (in ENG and JAM tactics) whether or not the player has already fired at the target during the lifetime of its current perceptions. When used with ASG tactics it checks if the current subordinate has fired at the target in the lifetime of its current perceptions.
- FIRING-NOW
  Tactics: ASG, ENG, JAM, LAUNCH, EMCON, GUNS.

  ```
  FIRING-NOW IS YES
  ```

  FIRING-NOW checks, with ENG tactics, if the current weapon is firing at a target now. With ASG, LAUNCH and GUNS tactics it checks if any relevant subordinate is firing at a target. Finally, with EMCON and JAM tactics it checks if any weapon belonging to the player itself is firing at the target. The meaning of 'currently firing at the target' depends on the type of weapon. If the weapon fires ordnance guided by the weapon, i.e. passively guided or controlled ordnance, then FIRING-NOW IS YES will be true from the decision to shoot until the intercept point. If the weapon's ordnance is uncontrolled, i.e. it is a simple projectile or self guiding, then the weapon is firing from the decision to fire until the weapon actually fires.
- FUEL-REMAINING
  Tactics: MOV.

  ```
  FUEL-REMAINING < 100.0 (KG)
  ```

  FUEL-REMAINING is used with MOVE-PLANS to test how much fuel a mover has left.
- GAME-TIME
  Tactics: ASG, ENG, JAM, LAUNCH, EMCON, GUNS.

  ```
  GAME-TIME < 4.0 (HR)
  ```

  GAME-TIME is the time reached in the scenario.
- HDG-CROSS-ANGLE
  Tactics: ASG, ENG, JAM, MOV.

  ```
  HDG-CROSS-ANGLE < 1.2 (RADIANS)
  ```

  HDG-CROSS-ANGLE tests the size of the heading of the target relative to the heading of the resource. It is only appropriate if both the target and the resource are moving.
- IFF-STATUS
  Tactics: ASG, ENG, JAM.

  ```
  IFF-STATUS IS HOSTILE
  ```

IFF-STATUS is used to confirm the 'Identification Friend or Foe' (IFF) status of a perceived target. The status will be one of HOSTILE, FRIEND, NEUTRAL or NOT-KNOWN.

- INTERCEPT-RESULTS-KNOWN
  Tactics: ASG, ENG, JAM.

```
INTERCEPT-RESULTS-KNOWN
```

INTERCEPT-RESULTS-KNOWN is true when none of the players (or subordinates for ASG tactics) are currently firing at a target. It is useful in LETHAL-ENGAGE-FIRING-START to delay a subsequent salvo until the results of a previous shot are known or in LETHAL-ASSIGNMENT-STOP to prevent cancelling an assignment whilst the subordinate is shooting at a target.

- JAM-CONTROL-MODE
  Tactics: ENG, ASG, EMCON, JAM, GUNS, LAUNCH.

```
JAM-CONTROL-MODE IS SELF
```

JAM-CONTROL-MODE allows a player to check if it has the authority to disrupt a target without receiving authority from a commander. The status of this mode is initially set in the SDB using the MODES-OF-CONTROL: item DISRUPT.

- JAMMER-STATUS
  Tactics: JAM.

```
JAMMER-STATUS IS JMR-ON
```

JAMMER-STATUS is used to see if the disruptor has status JMR-ON, JMR-OFF or JMR-NON/OP, i.e. on, off or non-operational. The status of the disruptor is initially set using the SYSTEM: item of the SDB.

- LAST-COMM-PICKUP
  Tactics: JAM.

```
LAST-COMM-PICKUP > 5.0 (MIN)
```

LAST-COMM-PICKUP gives the time since the last perception of the targeted emitter.

- LAST-SENSED
  Tactics: ASG, ENG, JAM, MOV.

```
LAST-SENSED < 30.0 (SEC)
```

LAST-SENSED tests the time elapsed since the last sensory perception of the target. If the player knows of the target through indirect intelligence this time is that since the most recent sensory detection occurred which then led to the indirect intelligence.

- LAUNCH-CONTROL-MODE
  Tactics: ENG, ASG, EMCON, JAM, GUNS, LAUNCH.

```
LAUNCH-CONTROL-MODE IS SELF
```

LAUNCH-CONTROL-MODE allows a player to check if it has the authority to launch a subordinate without receiving authority from a commander. The status of this mode is initially set in the SDB using the MODES-OF-CONTROL item LAUNCH.

- LAUNCHES-SO-FAR
  Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH.

```
LAUNCHES-SO-FAR NO-MORE-THAN 3 (VEHICLES)
  RE: interceptor RESOURCE
```

LAUNCHES-SO-FAR is used to count the number of resources already launched by the player. The count may be restricted to the types of PLAYERS or FUTURE-PLAYERS indicated in the associated RE: phrases. In the above example the named resource is 'interceptor'.

- MOVING-TO-ENGAGE
  Tactics: ASG, ENG, JAM.

```
MOVING-TO-ENGAGE IS NO
```

MOVING-TO-ENGAGE checks if the current target is the same as the target being pursued in the last reactive manoeuvre. A disaggregated player, i.e. a future player, shot at a target by a weapon will automatically be moving to engage that target.

- MY-ALT
  See RESOURCE-ALT
- MY-HDG
  See RESOURCE-HDG.
- MY-SPD
  See RESOURCE-SPD.
- NET-TYPE
  Tactics: JAM.

```
NET-TYPE IS some_net
```

NET-TYPE checks to see if the perceived communication transmitter is broadcasting on the named net: 'some_net' in the example above. The net's name will be given in either the EXPLICIT-NETS or IMPLICIT-NETS section of the UAN.

- OPERATIONAL-SUBS
  Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH.

```
OPERATIONAL-SUBS AT-LEAST 3 (PLAYERS)
```

OPERATIONAL-SUBS can be used to confirm the number of operational subordinates. A subordinate is operational if it is alive and still retains ammunition, i.e. weapon resources. This criterion may be qualified to specify specific players, for example:

```
OPERATIONAL-SUBS AT-LEAST 3 (PLAYERS)
  RE: fighter_player PLAYER-TYPE
```

would restrict consideration to subordinates of the 'fighter_player' player-type. A second qualifier may be used to specify if the player's own direct subordinates are to be used:

```
OPERATIONAL-SUBS AT-LEAST 3 (PLAYERS)
  RE: SELF REFERENCE-CMDR
```

or, for when OPERATIONAL-SUBS is used with ASG, LAUNCH and GUNS tactics, the criterion can refer to the subordinates of the current player's relevant subordinates:

```
OPERATIONAL-SUBS AT-LEAST 3 (PLAYERS)
  RE: SUB REFERENCE-CMDR
```

- OTHER-SYSTEMS-JAMMED-FOR
  Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH.

```
OTHER-SYSTEMS-JAMMED-FOR > 3.0 (MIN)
```

OTHER-SYSTEMS-JAMMED-FOR allows the status of sensor receivers or communication receivers other than those specified as the current resource to determine the current tactical response. In the above example, all sensor and communication receivers belonging to the current player will be checked to see if any of them have been jammed for more than three minutes. (The player must be aware that these systems are in fact being jammed using the J/N-NOISE-OPERATOR-THRESHOLD or J/N-PULSE-OPERATOR-THRESHOLD for sensors, and the J/N-COMM-OPERATOR-THRESHOLD for communication devices.) Specifically named sensor and communication receivers can be identified with the qualifiers SNR-TYPE and COMMO-TYPE respectively:

```
OTHER-SYSTEMS-JAMMED-FOR > 3.0 (MIN)
  RE: long_range_radar_rx SNR-TYPE
  RE: radio_rx COMMO-TYPE
```

- PERCEPTION
  Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH.

```
PERCEPTION IS DIRECT-INTELL
```

PERCEPTION tests the nature of the source of the target perception. In the above example the source tested is direct intelligence, i.e. the perception has been obtained from one of the player's own sensors. Alternatively, this option may be INDIRECT-INTELL in which case the source of the perception was intelligence received from another player.

The perception source may be qualified to determine the source of the information. If the perception satisfies the DIRECT-INTELL criterion the type of sensor which was used may be tested, for example:

```
PERCEPTION IS DIRECT-INTELL
  RE: acquisition_radar SNR_TYPE
```

For indirectly received intelligence, two qualifiers may be applied either singly or jointly: PLAYER-TYPE which constrains the type of the player which originated the perception and COMMO-TYPE which indicates the type of communication receiver used to detect the intelligence. An example is:

```
PERCEPTION IS INDIRECT-INTELL
  RE: aew_player PLAYER_TYPE
  RE: radio_rx COMMO_TYPE
```

which constrains the perception to originate with a player of the type 'aew_player' and to be received with the 'radio_rx' communication receiver.

- PERCEPTION-SOURCE
  Tactics: MOV.

```
PERCEPTION-SOURCE IS DIRECT-INTELL
```

PERCEPTION-SOURCE is equivalent to the PERCEPTION criterion but is used with the MOVE-PLANS tactics rather than the RESOURCE-ALLOCATION tactics. Its syntax is slightly different, the options for the source of the perception being one of DIRECT-INTELL, INDIRECT-INTELL, a named sensor type for direct intelligence or a named player-type for indirect intelligence. The type of communication receiver cannot be specified. The nearest equivalents to the above example for RESOURCE-ALLOCATION tactics would be:

```
PERCEPTION-SOURCE IS acquisition_radar
```

for direct intelligence and

```
PERCEPTION-SOURCE IS aew_player
```

for indirect intelligence.

- PERCEPTION-AGE
  Tactics: ASG, ENG, JAM.

```
PERCEPTION-AGE < 20.0 (MIN)
```

PERCEPTION-AGE checks the time elapsed since the current target was first perceived.

- REL-SUB-HDG
  Tactics: ASG, ENG, JAM, MOV.

```
REL-SUB-HDG > 25.0 (DEG)
```

  REL-SUB-HDG is the angle between the heading of the resource (or mover in MOVE-PLANS tactics) and the range vector drawn from the mover to the target. This angle, along with REL-TGT-HDG, is a two-dimensional angle obtained by measuring the relative directions along the ground. This criterion is only appropriate for moving targets.
- REL-TGT-ALT
  Tactics: ASG,ENG,JAM, MOV.

```
REL-TGT-ALT > 500.0 (M)
```

  REL-TGT-ALT specifies how much higher the target is than the resource or mover. The value of this threshold in MOVE-PLANS may be varied by the COMMIT-ALT table of the mover's capability block, see section 3.4.1.
- REL-TGT-HDG
  Tactics: ASG, ENG, JAM, MOV.

```
REL-TGT-HDG > 1.0 (RADIANS)
```

  REL-TGT-HDG is the angle between the heading of the target and the range vector drawn from the target to the mover.
- RELOAD-STATUS
  Tactics: ENG.

```
RELOAD-STATUS IS NO
```

  RELOAD-STATUS checks if a weapon is currently being reloaded.
- RESOURCE-ALT
  Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH.
  MY-ALT
  Tactics: MOV.

```
RESOURCE-ALT < 5000.0 (FT)
```

  RESOURCE-ALT gives the current altitude above MSL of the resource in RESOURCE-ALLOCATION tactics; MY-ALT the altitude of the mover in MOVE-PLANS.
- RESOURCE-HDG.
  Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH.
  MY-HDG
  Tactics: MOV.

```
RESOURCE-HDG > 45.0 (DEG)
```

  RESOURCE-HDG tests the current heading of the resource (in RESOURCE-ALLOCATION tactics); MY-HDG that of the mover (in MOVE-PLANS). The heading is measured anti-clockwise from due east.

- RESOURCE-SPD.
  Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH.
  MY-SPD
  Tactics: MOV.

```
RESOURCE-SPD < 100.0 (M/SEC)
```

RESOURCE-SPD checks the current speed of the resource (in RESOURCE-ALLOCATION tactics); MY-SPD tests the speed of the mover (in MOVE-PLANS).

- ROUNDS-FIRED-DURING-ENG
  Tactics: ENG.

```
ROUNDS-FIRED-DURING-ENG NO-MORE-THAN 10 (ROUNDS)
```

ROUNDS-FIRED-DURING-ENG is used to check the amount of ammunition that has been used since the current engagement commenced. It may be qualified to refer to a specific type of ordnance:

```
ROUNDS-FIRED-DURING-ENG AT-LEAST 6 (ROUNDS)
  RE: rockets (ORDNANCE)
```

- SALVOS-FIRED-DURING-ENG
  Tactics: ENG.

```
SALVOS-FIRED-DURING-ENG NO-MORE-THAN 3 (SALVOS)
```

SALVOS-FIRED-DURING-ENG checks the number of salvos that have been fired since the current engagement commenced. It may be qualified to refer to a specific type of ordnance:

```
SALVOS-FIRED-DURING-ENG AT-LEAST 2 (SALVOS)
  RE: missiles (ORDNANCE)
```

- SENSOR-STATUS
  Tactics: ENG, EMCON.

```
SENSOR-STATUS IS SNR-ON
```

SENSOR-STATUS is used in EMCON procedures to see if the sensor status is SNR-ON, SNR-OFF or SNR-NON/OP, i.e. on, off or non-operational. The status of the sensor is set initially using the SYSTEM: item of the SDB. With ENG procedures the status of the tracking sensor linked to the relevant weapon is checked.

- SKY-RADIANCE
  Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH.

```
SKY-RADIANCE IS DAY
```

SKY-RADIANCE tests the value of the option SKY RADIANCE set in the MOD. The options may be either DAY or NIGHT and is used in computing sensing chances made with optical and infrared sensor receivers.

- SNR-FREQ
  Tactics: JAM.

```
SNR-FREQ > 1.25 (MHZ)
```

SNR-FREQ evaluates the transmission frequency of the targeted transmitter.

- SNR-STATUS
  Tactics: MOV.

```
SNR-STATUS IS DETECT
```

SNR-STATUS is used with MOVE-PLANS tactics only to determine the status of any ongoing lethal engagements against the current target. Although SNR-STATUS is used only within MOVE-PLANS it will be ineffectual without the presence of lethal engagement (ENG) procedures in the player's RESOURCE-ALLOCATION block. Furthermore, note that SNR-STATUS should not be confused with the RESOURCE-ALLOCATION criterion SENSOR-STATUS, whose function is quite different.

The above example, SNR-STATUS IS DETECT, will be true when a lethal engagement is either proceeding or may potentially proceed against the current target. The alternative criterion is SNR-STATUS IS LOSE-DETECT which will be true if a lethal engagement is either being terminated or has recently being terminated against the target.

The SNR-STATUS criterion is an important means of co-ordinating a mover's reactive manoeuvres with a lethal engagement of a target.

- SUB-ENG-CONTROL-MODE
  Tactics: ASG, LAUNCH.

```
SUB-ENG-CONTROL-MODE IS SELF
```

SUB-ENG-CONTROL-MODE tests the lethal engage mode of control of the relevant subordinates. The lethal engage mode of control is initially set by the ENGAGE keyword of the MODES-OF-CONTROL: item in the SDB, but may be varied dynamically by the GUNS-FREE/GUNS-TIGHT tactics within the RESOURCE-ALLOCATION block. The test may be compared with either SELF, in which case the player making the evaluation has engagement control, (i.e. the commander), SUB in which case the subordinate has authority to engage, or a specific player name.

- SUB-STATUS
  Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH.

```
SUB-STATUS IS SUB-OP
```

SUB-STATUS is used to see if a subordinate is operational (SUB-OP) or out of action, (SUB-O/A) for ASG, GUNS and LAUNCH tactics. For ENG, JAM and EMCON tactics the test actually checks the status of the player itself. (A player may be damaged and out of action for particular activities but still able to use remaining systems for other roles.)

- SUBORDINATE-JAMMED-FOR

Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH.

```
SUBORDINATE-JAMMED-FOR > 30.0 (SEC)
  RE: interceptor PLAYER-TYPE
  RE: ANY SNR-TYPE
```

SUBORDINATE-JAMMED-FOR allows decisions to be made which allow for circumstances in which a subordinate has jammed sensors. The qualifiers referring to the player-types of the subordinates and their sensor receiver types are mandatory. However, the inclusion of the ANY keyword in both qualifiers would be equivalent to their absence.

The criterion tests the maximum time any of the relevant sensors have been jammed for, assuming that the jamming has been recognized, (i.e. the jamming signal has exceeded the J/N-NOISE-OPERATOR-THRESHOLD or the J/N-PULSE-OPERATOR-THRESHOLD).

In the LAUNCH, GUNS and ASG tactics the criterion may be made to refer to the subordinates of the relevant subordinates of the current player, with the use of the RE: SUB REFERENCE-CMDR qualifier:

```
SUBORDINATE-JAMMED-FOR > 5.0 (MIN)
  RE: ANY PLAYER-TYPE
  RE: long_range_radar_rx SNR-TYPE
  RE: SUB REFERENCE-CMDR
```

The default value of the REFERENCE-CMDR option is SELF, which means that the subordinates referred to are the direct subordinates of the current player.

- SYSTEM-STATUS

Tactics: ASG, ENG, JAM, EMCON, GUNS, LAUNCH.

```
SYSTEM-STATUS IS OPERATIONAL
  RE: tracking_radar_rx LINKED-TRACKER
```

SYSTEM-STATUS checks if the appropriate linked tracking sensor receivers are operational. The types of the tracking sensor receivers must be identified in the criterion's qualifying statement. Which receivers are checked depends on which tactical block the criterion is included within. In all cases the status will be OPERATIONAL if at least one such functional tracking sensor receiver is found. For lethal engagement tactics, ENG, all trackers of the appropriate type linked to the current weapon are checked. With assignment tactics, ASG, LAUNCH and GUNS, the trackers linked to all weapons belonging to the relevant subordinates are checked. For emission control, EMCON, and disruptor tactics, JAM, the status of the trackers linked to all weapons on the current player are checked.

- TGT-ALT
  Tactics: ASG, ENG, JAM, MOV.

```
TGT-ALT > 1000.0 (M)
```

  TGT-ALT tests the height above MSL of the target.
- TGT-ASPECT-ANGLE
  Tactics: MOV.

```
TGT-ASPECT-ANGLE < 20.0 (DEG) LEFT
```

  TGT-ASPECT-ANGLE tests the angle between the range vector drawn from the mover to the target and the heading of the target. Suppressor always takes this angle to be positive and determines its sense by specifying whether or not the target's heading lies to the LEFT or RIGHT of the range vector. If the heading's sense is not important then the ABS keyword can be used to indicate that only the magnitude of the angle is important. This criterion is only used with MOVE-PLANS tactics.
- TGT-HDG
  Tactics: ASG, ENG, JAM, MOV.

```
TGT-HDG < 10.0 (DEG)
```

  TGT-HDG checks the current heading of a target. If the target is stationary its heading is assumed to be due east.
- TGT-SPD
  Tactics: ASG, ENG, JAM, MOV.

```
TGT-SPD > 400.0 (KNOTS)
```

  TGT-SPD tests the current speed of the target.
- TGT-TYPE
  Tactics: MOV.

```
TGT-TYPE IS sam_battery
```

  TGT-TYPE compares the current target type to a specific element name, for example, 'sam_battery' in the above example. It is only used in MOVE-PLANS tactics.
- TIME-LAPSE
  Tactics: MOV.

```
TIME-LAPSE > 30.0 (MIN)
```

  TIME-LAPSE is used within MOVE-PLANS tactics to test the time elapsed since either a repeating PATTERN was begun or the mover was brought to a standstill using the EXECUTE SUSPEND MOVEMENT action statement of the SDB. It is not appropriate for a non-repeating PATTERN.
- TIME-SEPARATION
  Tactics: ASG, ENG, JAM, MOV.

```
TIME-SEPARATION < 50.0 (SEC)
```

TIME-SEPARATION is the expected time to go before the intercept of the mover or resource and the target. It is computed assuming that both maintain current speed and heading. It will be effectively infinite when no intercept is possible.

- TIME-SINCE-STATUS-CHANGE
  Tactics: EMCON.

```
TIME-SINCE-STATUS-CHANGE > 0.75 (HR)
```

TIME-SINCE-STATUS-CHANGE is the time that has elapsed since the last change in the status of a sensor system, i.e. a change between SNR-ON, SNR-OFF and SNR-NON/OP. The above form will consider the time since the last change for the sensor receiver being currently considered by the EMCON procedures.

The criterion may be used in any RESOURCE-ALLOCATION tactical block if the sensor-receiver can be *uniquely* identified by its type, for example:
Tactics: ASG, ENG, JAM, LAUNCH, EMCON, GUNS.

```
TIME-SINCE-STATUS-CHANGE < 52.5 (SEC)
  RE: missile_tracking_sensor_rx SNR-TYPE
```

Since the first matching sensor receiver will be selected, this criterion should not be used when more than one sensor-receiver is of the given type.

- TOTAL-APPROACHING-TARGETS
  Tactics: ASG, ENG, JAM, LAUNCH, EMCON, GUNS.

```
TOTAL-APPROACHING-TARGETS AT-LEAST 3 (TGTS)
  RE: ANY TGT-TYPE
```

TOTAL-APPROACHING-TARGETS counts the number of targets which are closing with the current resource. The targets may be stationary as long as the relative velocity of the resource and the target bring the players closer together.

At least one TGT-TYPE qualifier identifying the target's possible player-types must be given, but it can use the keyword ANY to allow any target to be considered. Further qualifiers restricting attention to one or more zones may be associated with each TGT-TYPE entry, for example:

```
TOTAL-APPROACHING-TARGETS AT-LEAST 2 (TGTS)
  RE: interceptor TGT-TYPE
    RE: sam_zone_1 ZONE
    RE: sam_zone_2 ZONE
  RE: sam_player TGT-TYPE
    RE: ANY ZONE
```

associates different zones with different target types. The zones always belong to the player evaluating the tactics.

- TOTAL-ASGS

  Tactics: ASG, LAUNCH, GUNS.

```
TOTAL-ASGS AT-LEAST 4 (TGTS)
```

TOTAL-ASGS checks a quantity *one more* than the number of assignments made to the current subordinate (ASG) or all subordinates (GUNS, LAUNCH). The above example is identical to:

```
CURRENT-ASGS AT-LEAST 3 (TGTS)
```

- TOTAL-ENG'S

  Tactics: ENG, ASG, EMCON, JAM, GUNS, LAUNCH.

```
TOTAL-ENG'S NO-MORE-THAN 2 (TGTS)
```

TOTAL-ENG'S checks a quantity *one more* than the number of current engagements for the current weapon system, (ENG); the current subordinate or subordinates (ASG, LAUNCH, GUNS); or for the thinker's player (EMCON, JAMMER). The above example is identical to:

```
CURRENT-ENG'S NO-MORE-THAN 1 (TGTS)
```

- TOTAL-JAMMERS

  Tactic: ENG, ASG, EMCON, JAM, GUNS, LAUNCH.

```
TOTAL-JAMMERS NO-MORE-THAN 5 (SYSTEMS)
  RE: OFF JMR-STATUS
  RE: ANY JMR-TYPE
```

TOTAL-JAMMERS counts the number of disruptors which belong to the current player which have both the required type and status. One each of the two qualifiers must be given, JMR-TYPE may be either the keyword ANY or the name of any disruptor type. The JMR-STATUS is one of ANY, ON, OFF or NON-OP. Setting ANY in both qualifying statements would be functionally equivalent to omitting these items.

- TOTAL-SENSORS

  Tactic: ENG, ASG, EMCON, JAM, GUNS, LAUNCH.

```
TOTAL-SENSORS AT-LEAST 3 (SYSTEMS)
  RE: ON  SNR-STATUS
  RE: ANY SNR-TYPE
```

TOTAL-SENSORS counts the number of sensors which belong to the current player which have both the required type and status. One each of the two qualifiers must be given, SNR-TYPE may be either the keyword ANY or the name of any sensor receiver type. The SNR-STATUS is one of ANY, ON, OFF or NON-OP. Setting ANY in both qualifying statements would be functionally equivalent to omitting these items.

- TOTAL-SPOTS

  Tactic: JAM.

```
TOTAL-SPOTS AT-LEAST 3 (TGTS)
```

TOTAL-SPOTS checks a quantity *one more* than the number of spots being used by the current jammer. The above example is identical to:

```
CURRENT-SPOTS AT-LEAST 2 (TGTS)
```

- TOTAL-TARGETS

  Tactics: ASG, ENG, JAM, LAUNCH, EMCON, GUNS, MOV.

```
TOTAL-TARGETS NO-MORE-THAN 2 (TGTS)
  RE: ANY TGT-TYPE
```

TOTAL-TARGETS counts the number of targets for the current resource or mover. For resource allocation tactics the syntax is identical to TOTAL-APPROACHING-TARGETS, except that now any target may be considered and not just those which are closing. So, the example below uses the TGT-TYPE and ZONE qualifiers in a manner equivalent to TOTAL-APPROACHING-TGTS:

```
TOTAL-TARGETS AT-LEAST 2 (TGTS)
  RE: interceptor TGT-TYPE
    RE: sam_zone_1 ZONE
    RE: sam_zone_2 ZONE
  RE: sam_player TGT-TYPE
    RE ANY ZONE
```

TOTAL-TGTS may also be used within MOVE-PLANS with a slightly different syntax, the number of targets must be given as a real number and tested with the greater-than '>' and less than '<' symbols. So the above example becomes:

```
TOTAL-TARGETS > 1.5 (TGTS)
  RE: interceptor TGT-TYPE
    RE: sam_zone_1 ZONE
    RE: sam_zone_2 ZONE
  RE: sam_player TGT-TYPE
    RE ANY ZONE
```

- TOTAL-TIME

  Tactics: MOV.

```
TOTAL-TIME > 20.0 (MIN)
```

TOTAL-TIME is used within MOVE-PLANS tactics to test the time elapsed since a mover first started to move.

- TRACKING-STATUS
  Tactics: ENG, EMCON.

```
TRACKING-STATUS IS TRACKING
```

TRACKING-STATUS checks the current status of a tracking sensor receiver; it may be one of NOT-APPLICABLE, ATTEMPTING-TRACK, TRACKING, COASTING or FIRING. When used with lethal engagement tactics, ENG, it looks at a tracker linked to the current weapon resource. In this case it may be qualified to identify a tracker of a specific sensor type, for example:

```
TRACKING-STATUS IS COASTING
  RE: fire_control_rx SNR-TYPE
```

If more than one sensor receiver matches the constraints the one whose status is closest to FIRING will be selected.

When used with EMCON tactics then the current sensor receiver resource is relevant and the qualifying statement should not be used.

- VEHICLES-LEFT
  Tactics: ASG, LAUNCH, GUNS.

```
VEHICLES-LEFT NO-MORE-THAN 2 (VEHICLES)
  RE: interceptor RESOURCE
```

VEHICLES-LEFT counts the number of vehicles of the indicated type, either a player or future-player, which may still be launched by the current player.

- WPN-STATUS
  Tactics: ENG, ASG, JAM, LAUNCH, GUNS, EMCON.

```
WPN-STATUS IS WPN-OP
```

WPN-STATUS checks that a weapon system is operational, WPN-OP, or non-operational, WPN-NON/OP. In lethal engagement procedures, ENG, the weapon tested is that currently selected as the resource. With assignment procedures, ASG, LAUNCH and GUNS, all weapons associated with the relevant subordinates are checked to see if at least one is operational (or not). The number of weapons chosen can here be restricted with a qualifying RESOURCE statement, e.g.

```
WPN-STATUS IS WPN-OP
  RE: missile_launcher RESOURCE
```

For emission control, EMCON, and disrupting tactics, JAM, all weapons that belong to the current player are evaluated, although again the weapons selected can be restricted with a qualifying statement.

- 2D-CLOSING-SPD

  Tactics: ASG, ENG, JAM, MOV.

  ```
  2D-CLOSING-SPD > 100.0 (MPH)
  ```

  2D-CLOSING-SPD is the relative ground speed of the resource and the target, and is obtained by projecting their velocities along the range vector separating them. It is positive when their locations are closing, negative when receding.
- 2D-DIST

  Tactics: ASG, ENG, JAM.

  ```
  2D-DIST < 5.0 (KM)
  ```

  2D-DIST-TO-TGT

  Tactics: MOV.

  ```
  2D-DIST-TO-TGT < 23.0 (MILES)
  ```

  2D-DIST or, for MOVE-PLANS tactics, 2D-DIST-TO-TGT tests the ground distance between the resource and target.
- 2D-DIST-TO-INT

  Tactics: MOV.

  ```
  2D-DIST-TO-INT < 23.0 (MILES)
  ```

  2D-DIST-TO-INT tests the ground distance to the computed intercept point of the mover and target.
- 2D-REL-TGT-OFFSET

  Tactics: ASG, ENG, JAM, MOV.

  ```
  2D-REL-TGT-OFFSET > 1000.0 (M)
  ```

  2D-REL-TGT-OFFSET checks the distance from the resource to the extrapolated point of closest approach of the target, assuming the target always moved at its current speed and the resource was still. This value is an instantaneous measure of the separation of the mover and the target and is not meant to be predictive. It only has meaning for moving targets.
- 2D-REL-TGT-UP/DOWN-RANGE

  Tactics: ASG, ENG, JAM, MOV.

  ```
  2D-REL-TGT-UP/DOWN-RANGE < -0.0 (M)
  ```

  2D-REL-TGT-UP/DOWN-RANGE is the ground distance from the target to the anticipated point of closest approach referred to by 2D-REL-TGT-OFFSET. It is negative if the target is heading away from this point and positive if heading towards it. This term is a measure of whether or not a target is moving towards or away from the resource. The same measure can be used as part of a weapon's P_K table as part of the RNG dimension in WPN-PK.

- 3D-DIST
  Tactics: ASG, ENG, JAM.
  ```
  3D-DIST > 15.0 (KM)
  ```
  3D-DIST-TO-TGT
  Tactics: MOV.
  ```
  3D-DIST-TO-TGT < 2000.0 (FT)
  ```
  3D-DIST or, for MOVE-PLANS tactics, 3D-DIST-TO-TGT tests the true three-dimensional distance between the resource and the target.
- 3D-TGT-LOC
  Tactics: ASG, ENG, JAM, MOV.
  ```
  3D-TGT-LOC WITHIN sam_zone
  ```
  3D-TGT-LOC tests whether the target is located within or outside a named zone which by default is associated with the resource or mover. When used with RESOURCE-ALLOCATION tactics, i.e. ASG, ENG and JAM, the zone may also be referred to the player making the decision with the YOUR-LOC option of the REFERENCE-LOC qualifier:
  ```
  3D-TGT-LOC OUTSIDE sam_zone
    RE: YOUR-LOC REFERENCE-LOC
  ```
  The default option to this qualifier is SUB-LOC.

It can be seen that great flexibility can be enjoyed in designing the tactics used by Suppressor players. Many particular instances can be tested for, such as the location, number and condition of the targets, the availability and status of the player's resources and subordinates and so on. In the following two sections how these criteria may be used to define the tactics of each player are discussed, beginning with the RESOURCE-ALLOCATION tactics and concluding with the MOVE-PLANS tactics.

3.6.2 The Tactics of Resource Allocation

The tactical criteria discussed above control only part of the process going on in a player's tactical evaluation of a situation. These criteria describe *what* each player thinks about when evaluating a decision. In addition to this, each player must also know both *when* and *how* to think about something. The latter of these, the mechanism of the thinking process, the 'how', is controlled by the capabilities of the player's thinkers. The timing of the thinking process, the 'when', is partly controlled by various evaluation rates which may be set for each tactical evaluation. A larger factor determining the timing of thinking processes is though the pressure of external events, i.e. the arrival of stimuli and the availability of thinkers to service each player's pending queue[18].

The ability to process each `RESOURCE-ALLOCATION` tactic or `MOVE-PLANS` tactic is specifically conferred on a thinker by the appropriate setting of the thinker's capabilities. For example in order for a thinker to consider reactive movement the thinking time `CONSIDER-MOVE` must be defined. Similarly other thinking times may be associated with each of the possible tactical responses invoked by the `RESOURCE-ALLOCATION` tactics. The thinking time `EVAL-LETHAL-ENGAGE` enables a thinker to process the tactics described by the `RESOURCE-ALLOCATION` tactics which start and stop lethal engagements, namely `LETHAL-ENGAGE-START` and `LETHAL-ENGAGE-STOP`.

The various types of `RESOURCE-ALLOCATION` tactics plus the `MOVE-PLANS` tactics which describe reactive manoeuvres constitute each player-type's reactive abilities. The mental processing of these tactics by thinkers is controlled initially by the appropriate events being entered on each player's pending queue and by the subsequent stimuli received by each player. Suppose for example a player is assigned to engage a specific target. Initially the target is beyond the player's sensing range, and so although the player is aware of the target it cannot yet see it. At some point the target is first detected and (presumably) added to a list of targets which the player may decide to engage if the conditions are right. This list is the 'lethal engagement queue' and may contain other candidates if other potential targets have also been detected. The addition and deletion of each target from this queue is controlled by the tactics in the `LETHAL-ENGAGE-QUEUE-ADD` and `LETHAL-ENGAGE-QUEUE-DROP` procedures respectively. Once a target is initially detected new information will be arriving from the player's sensors at a relatively high rate, describing such things as the position and speed of the target. At this point it would be impractical to allow the player's thinkers to evaluate the lethal engagement procedures every time new sensory information arrives. Instead a user defined evaluation rate is employed, the `ENG-EVAL-RATE`, which determines the maximum rate at which each player's thinkers will consider further actions against this target. These actions will either be to commence an engagement, a decision determined by the `LETHAL-ENGAGE-START` procedure, or

[18] The pending queue and its interaction with the player's thinking processes and perceptions was briefly discussed in section 2, 'Overview of Suppressor'.

delete the target from the lethal engagement queue according to the criteria of the LETHAL-ENGAGE-QUEUE-DROP procedure.

The selection of evaluation rates forms part of the TACTIC block for each player, for example the 'aew_player' has the following evaluation rates defined for it:

```
$ The outer shell of the AEW tactic block
TACTIC aew_tactics
$ Define the AEW craft's tactical evaluation rates
  EVALUATION-RATES
    ASG-EVAL-RATE         0.10  (1/SEC)
    EMCON-EVAL-RATE       0.20  (1/SEC)
    LAUNCH-EVAL-RATES     0.15  (1/SEC)
    FREE/TIGHT-EVAL-RATE  0.10  (1/SEC)
  END EVALUATION-RATES
  <Omitted tactics>
END TACTIC
```

Since the 'aew_player' has no weapons of its own, nor any capabilities for disrupting other systems its evaluation rates are associated with assignments, emission control, launching subordinates and changing subordinates' lethal modes of control respectively. The rates given are the maximum possible rates at which the evaluations will occur. So for example, a new lethal assignment evaluation will be scheduled 10 s after the end of the current evaluation, since the ASG-EVAL-RATE is 0.1 s^{-1}. The actual evaluation may well occur later, if the player's thinkers are busy and the pending queue is backlogged. Extra evaluation rates, ENG-EVAL-RATE and JAM-EVAL-RATE, can be used to control the frequency of lethal engagement and non-lethal engagement conditions.

The tables first introduced in the thinker's CAPABILITY blocks to associate thinking times with reactive mental processes in fact list all the possible RESOURCE-ALLOCATION procedures along with MOVE-PLANS. These tables are reproduced below but now also include the evaluation rates, which are relevant to some or all of the procedures within each grouping:

- Lethal-Assignments: ASG-EVAL-RATE and ASG-TGT-UPDATE-RATE

Thinking Time	RESOURCE-ALLOCATION procedure
EVAL-ASSIGN-THREAT	LETHAL-ASSIGNMENT-QUEUE-ADD LETHAL-ASSIGNMENT-QUEUE-DROP
CONSIDER-ASG/CANCEL	LETHAL-ASSIGNMENT-START LETHAL-ASSIGNMENT-STOP

- Lethal Engagements: `ENGAGE-EVAL-RATE`

Thinking Time	RESOURCE-ALLOCATION procedure
EVAL-ENGAGE-THREAT	LETHAL-ENGAGE-QUEUE-ADD LETHAL-ENGAGE-QUEUE-DROP
EVAL-LETHAL-ENGAGE	LETHAL-ENGAGE-START LETHAL-ENGAGE-STOP
EVAL-FIRING	LETHAL-ENGAGE-FIRING-START LETHAL-ENGAGE-FIRING-STOP

- Non-lethal Engagements: `JAM-EVAL-RATE`

Thinking Time	RESOURCE-ALLOCATION procedure
EVAL-JMR-QUEUE	JAMMER-QUEUE-ADD JAMMER-QUEUE-DROP
EVAL-JMR-SPOTS	JAMMER-SPOT-ADD JAMMER-SPOT-DROP

- Emission Control: `EMCON-EVAL-RATE`

Thinking Time	RESOURCE-ALLOCATION procedure
EVAL-EMCON-CHANGE	EMCON/TURN-ON EMCON/TURN-OFF

- Lethal Mode of Control: `FREE/TIGHT-EVAL-RATE`

Thinking Time	RESOURCE-ALLOCATION procedure
EVAL-GUNS-FREE/TIGHT	GUNS-FREE GUNS-TIGHT

- Launching Movement: `LAUNCH-EVAL-RATE`

Thinking Time	RESOURCE-ALLOCATION procedure
CONSIDER-LAUNCH	LAUNCH-START

The above procedures are divided into two classes according to whether or not the tactics are designed to deal with targets on one-by-one basis or to respond with an action which is not linked to any single target. For example a player representing a SAM missile system may have several missile batteries, each one of which is represented by a separate element which may engage targets independently of the other elements. The procedural groupings which allow particular targets to be selected are those describing lethal assignment, and both lethal and non-lethal engagement. In all these cases the procedures describe actions which ultimately relate to a particular target, although several targets may be being considered at once by a player's thinkers. For these procedures Suppressor uses a set of 'queues' in order to organize each player's actions. In the case of lethal engagement the queue is used to list the names of each candidate for lethal engagement. For non-lethal engagements the names of each of the communication or radar receivers which could currently be disrupted are entered on the queue. The lethal assignment queue contains not the names of potential targets but of the subordinates which are candidates for lethal assignments against perceived targets. In all cases the controlling logic is similar, and may be described with reference to the lethal engagement procedures, which are the most complex. Here target candidates are added and deleted from the appropriate queue by the `LETHAL-ENGAGE-QUEUE-ADD` and the `LETHAL-ENGAGE-QUEUE-DROP` procedures Once a target is in the queue the conditions which define when an engagement may actually proceed are given in `LETHAL-ENGAGE-START` procedure; similarly, the `LETHAL-ENGAGE-STOP` procedure defines when the engagement should cease. During a lethal engagement of a target the procedures `LETHAL-ENGAGE-FIRING-START` and `LETHAL-ENGAGE-FIRING-STOP` determine when weapons will both commence and cease fire against a target. Each relevant procedure within these three classes will be evaluated once for each target in a cycle whose minimum length is set by the appropriate evaluation rate.

The remaining procedural groupings, dealing with emission control, launching movement and lethal mode of control are somewhat different. These represent definitive actions of a player with regard to its resources which must either be taken or not. For example, consider again the player representing a SAM site with several missile batteries each having an associated tracking radar. Now whilst each battery may have within its range several possible targets each of the radar sensors must either be on or off; they cannot be at once turned on for one target and off for another. Therefore it would make no sense to consider the emission control tactics on a target by target basis; instead a single decision pertaining to the total situation must be made at the relevant moments. Similarly, in the movement and lethal mode of control groupings resources will either be employed or not; they cannot be variously allocated to different targets simultaneously. In these procedures, queues of candidates are not needed and the resulting tactics are therefore simpler. Furthermore, each procedure in these classes will only be evaluated once in each evaluation cycle, regardless of the number of potential targets. Note that the decisions made according to the tactics in these blocks, although not explicitly dependent on individual targets, might still be implicitly dependent on them. For example, the decision to assign a radar to track a target in a lethal engagement procedure could result in an emission control procedure activating the sensor.

Unlike other blocks of player-type tactics several RESOURCE-ALLOCATION blocks can be introduced within the TACTIC block, although each procedure may only be defined once. It is therefore possible to group procedures functionally within separate RESOURCE-ALLOCATION tactical blocks, which is a useful feature when the length and complexity of the tactics involved in these blocks is considered. The procedures themselves are identified by just a one line header containing the procedure's name. An example of this is:

```
$ Define the tactics controlling a bomber player-type
TACTIC bomber_tactics
  <Omitted tactics>
  RESOURCE-ALLOCATION
$   Define lethal engagement procedures
    LETHAL-ENGAGE-QUEUE-ADD
      <Omitted LETHAL-ENGAGE-QUEUE-ADD criteria>
    END LETHAL-ENGAGE-QUEUE-ADD
    <Omitted lethal engagement procedures>
  END RESOURCE-ALLOCATION
  <Omitted RESOURCE-ALLOCATION procedures>
END TACTIC
```

In the above, a single RESOURCE-ALLOCATION block is used to hold the procedures relevant to the player's lethal engagement procedures. These may be specified in any order, with the first here being LETHAL-ENGAGE-QUEUE-ADD. This procedure, which determines which perceived players may be added to a list of candidate targets, is terminated by the END LETHAL-ENGAGE-QUEUE-ADD statement. For these tactics to be useful all six of the lethal engagement procedures must be defined for this player-type. Similarly, when any tactical function, such as lethal assignment, is required all the lethal assignment procedures must be included in the player-type's tactics and so on.

Procedural structure

All RESOURCE-ALLOCATION procedures follow a standard format, and as noted above they are identified by their names within each RESOURCE-ALLOCATION block. This section describes the basic structure of each procedure using specific examples to illustrate the format. Any deviations or specific restrictions to this structure which are appropriate to specific procedures will be indicated in the relevant sections. For example, not all the possible forms of the target paragraph identifier can be used in all the procedures, and this will be indicated in the sections dealing with the specific procedures.

Each procedure consists of at least one 'target paragraph' which describes the nature of the targets being considered. For example, a target paragraph might identify a particular player-type as a target. Each paragraph is made up of a set of logical criteria which are used to determine if a potential target is suitable for further action. These criteria check such conditions as the distance of the target, whether it is in fact hostile or not, its speed and so on. If a target is selected by the criteria the final portion of the paragraph will attempt to allocate a resource to use against it, or to de-allocate a resource that is already assigned. For example a lethal assignment procedure might assign a subordinate to attack a target.

Target paragraphs commence with the TGT-TYPE keyword followed by an identifier specifying what constitutes a target in the current paragraph. So the identifier:

```
TGT-TYPE ANYONE
```

means that the following paragraph refers to any player in the scenario. A single player-type may be specified as a potential target, so that:

```
TGT-TYPE sam_player
```

specifies just players of the type 'sam_player' as potential targets. An even more restrictive example is:

```
TGT-TYPE sam_player WITH-ELEMENT sam_battery END
```

which specifies particular elements of a single player-type as legitimate targets. This option is sometimes necessary against players which have multiple locations to ensure that the correct part of the player is targetted.

Lists of names can be used to qualify both the TGT-TYPE and the WITH-ELEMENT identifiers and WITH-ELEMENT can be used to qualify the ANYONE qualifier:

```
TGT-TYPE ANYONE
  WITH-ELEMENT sam_battery  long_range_radar_tx  END
```

In the above example any player with SAM batteries or long range radar transmitters might be targetted.

Further possibilities exist for target identification. Targets may be explicitly excluded with the EXCEPT qualifier, so that:

```
TGT-TYPE ANYONE EXCEPT fighter_player
```

would target any player except those of the player-type 'fighter_player'.

In procedures with more than one target paragraph the final paragraph can be used as a 'default' paragraph to catch targets not otherwise specified by using the `ALL-OTHERS` identifier:

```
TGT-TYPE ALL-OTHERS
```

The `ALL-OTHERS` identifier can also be qualified with either the `WITH-ELEMENT` or `EXCEPT` options.

Since each potential target may be identified by several target paragraphs with conflicting results Suppressor will only allow each candidate target to be considered by the criteria in one target paragraph. To select which paragraph is used the sequencing of the target paragraphs is done in a specific order with the first matching paragraph being the one used. The order used is:

1. `TGT-TYPE ANYONE`
2. `TGT-TYPE target_name`
3. `TGT-TYPE ALL-OTHERS`

If more than one paragraph of each type occurs then they will be considered in the sequence they occur in the procedure. Note that if multiple target paragraphs with identical target identifiers do occur in the procedure then only the first one will ever be used. Because more than one paragraph may refer to a single target the user should always be careful that the desired paragraph is in fact being used to process each target.

Once a candidate has been matched with a target paragraph the next step is to filter out targets against whom no action will be taken. For example both sides in a scenario may have players of the identical type, e.g. F16 fighter aircraft. Since both sets of aircraft will match the target identifier the criteria within the target paragraph must distinguish them. Furthermore it must decide if action against a hostile F16 is worthwhile; the questions that might be asked are is it within range and can the player in fact act against the F16?

The identification of the targets is accomplished by a set of 'filters'. These filters are simply logical criteria made up of building blocks which are conditional statements. The filters plug together to act like one or more pipes, input data entering at the top passing through one or more filters with the final, filtered, output data leaving at the bottom. The data entering at the top are the candidate targets matching the target paragraph's identifiers; the data leaving the bottom are the targets which have passed all the filters making up the pipes. These targets are then passed to a selection stage, the final portion of each target paragraph, where resources are allocated to respond to them. In the case of lethal assignments these resources are the subordinates which can be assigned to engage the targets. No resources may be available if all are busy so the final stage of the process might fail.

Each filter consists first of an entry identifying the origin of the filter's input data. These data are the names of candidate targets and the first filter in each pipe accepts data from the player's perceptions, identified with the `INPUT` qualifier:

```
USE INPUT FOR FILTER 1
```

The above means that `FILTER 1` uses data input from the environment, and is thus the head of a pipe of filters. (Each filter in a procedure should be numbered uniquely with a positive integer, but there are no other restrictions on these numbers.)

The next filter in this pipe, should there be one, will accept input from the first filter using the `FILTER 1 SELECTIONS` qualifier:

```
USE FILTER 1 SELECTIONS FOR FILTER 2
```

so that `FILTER 2` reads data from `FILTER 1`. A pipe may fork so that more than one filter may read the output of a single filter. For example a third filter may exist in parallel to the second and so also take data from the first filter:

```
USE FILTER 1 SELECTIONS FOR FILTER 3
```

The first entry in each filter identifies the nature of the resource available to the procedure, for lethal assignments these resources are subordinates. These are indicated with the `SUB-TYPE` identifier followed by a list of the available subordinates' names. For example, if the third filter is used to allocate subordinates of player-type 'fighter_player' then it would begin:

```
USE FILTER 1 SELECTIONS FOR FILTER 3
  SUB-TYPE fighter_player
```

The resource type appropriate for each class of procedure is given in the following table:

Tactical Type	Resource Type
Lethal Assignment	`SUB-TYPE`
Lethal Engagement	`WPN-TYPE`
Non-Lethal Engagement	`JAMMER-TYPE`
Emission Control	`SNR-TYPE`
Lethal Mode of Control	`SUB-TYPE`
Launching Movement	`SUB-TYPE`

When the resource type is a subordinate then the names of the resources will be those of player-types, for all other resource types they will be names of the appropriate systems. So, for example, in lethal engagement procedures the resources are weapons and so will have type `WPN-TYPE`.

The main body of each filter is a set of criteria which are used to select whether the candidate target may be passed for possible future selection. These criteria are drawn from the list of possible criteria which were earlier described in detail. The criteria may be conjoined with either a logical AND statement or a logical OR statement. Since the AND conjunction has priority over the OR conjunction the compound construction of three criteria A, B and C:

```
A AND B OR C
```

will be true if C is true or if both A and B are true. In other words the above is equivalent to:

```
(A AND B) OR C
```

(However, note that parentheses must not be used in Supressor). If the alternative interpretation of the statement is required to be true, namely if both A and either B or C is true, that is if:

```
A AND (B OR C)
```

is required this cannot be accomplished with a single filter. Instead two filters should be used in series, the first testing that A is true and the second that either B or C is true.

The final filter in any pipe feeds its output into a statement which make the final selection of an appropriate resource. An example of this is:

```
FROM FILTER 2 SELECTIONS
  CHOOSE-FROM fighter_player
  PICK-AT-MOST 2 NOW 6 TOTAL
```

Here filter 2 was the last filter in a pipe and any candidate target which passes this filter will be eligible to have the subordinate 'fighter_player' allocated against it. (In procedures such as LETHAL-ASSIGNMENT-STOP or LETHAL-ENGAGE-QUEUE-DROP where processes are being terminated the resources are actually de-allocated.)

The appropriate resources are identified by the CHOOSE-FROM keyword, for example:

```
CHOOSE-FROM fighter_player
```

The CHOOSE-FROM keyword can be qualified with several different commands, which vary considerably from procedure to procedure. For example, all lethal assignment procedures may qualify the CHOOSE-FROM keyword with either WITH-TGT-CUING-FOR-LOC or WITH-SDB-CUING-FOR-LOC. In Suppressor parlance 'cuing' means that a resource's heading is pointed towards a specified position. Hence suppose we had the statement

```
CHOOSE-FROM sam_player
  WITH-TGT-CUING-FOR-LOC 2
  WITH-SDB-CUING-FOR-LOC 3
```

Then this would mean that the headings of the second and third locations of the 'sam_player' will be set. (The 'sam_player' has been referred to above, it has two 'sam_battery' elements at locations 2 and 3 respectively, and a headquarters element at location 1.) The element '20 sam_battery' at location 2 will be cued towards the perceived position of the target, (which might or might not be the target's real position), whereas '30 sam_battery' at location 3 will be cued to the heading specified in the HDG: item of the SDB. (If this latter quantity is not defined then the battery will have the default heading of due east set.) Note that cuing should not be used for moving locations, (such as the 'fighter_player' subordinate), since the heading of these locations is aligned with the velocity vector of the mover.

The following table lists all the possible qualifying phrases and indicates with which RESOURCE-ALLOCATION procedures they can be used. A bullet symbol, '•', indicates the qualifying statement may be used with all procedures in the appropriate tactical class. In the lethal engagement tactics those qualifiers which can only be used with LETHAL-ENGAGE-START are shown with an 'S' and those that can only be used with LETHAL-ENGAGE-FIRING-START with an 'F'. The four qualifying phrases that are required in certain tactical procedures are indicated with an 'R'. No qualifiers can be used with the non-lethal engagement procedures, JAM.

	ASG	ENG	GUNS	LAU	EMC	JAM
WITH-TGT-CUING-FOR-LOC	•	•				
WITH-SDB-CUING-FOR-LOC	•	•	•		•	
WITH-TRACKER		S				
WITH-TRACKER-GROUP		S				
WITH-ORDNANCE		F,R				
WITH-ELEMENT		F,R				
WITH-SALVO-SIZE		F				
WITH-PLAN		F		•,R		
WITH-VEHICLE				•,R		

The roles of the various qualifiers are described below:

- WITH-TRACKER
  LETHAL-ENGAGE-START

```
WITH-TRACKER short_range_tracker_rx
```

WITH-TRACKER may only be used in the LETHAL-ENGAGE-START procedure. When used, Suppressor will attempt to find a resource with a linked operational tracking receiver of the named type. Such a receiver must not have status SNR-NON/OP but may be currently turned off, i.e. have status SNR-OFF. Furthermore, the receiver must have spare tracking capacity, i.e. not already tracking its maximum number of targets as defined by MAX-PARALLEL-TRACKS. Finally, the tracker must be able to detect the current target, i.e. the tracker must be present in the target's SNR-ELE-INTERACTIONS.

More than one WITH-TRACKER option can be specified, in which case Suppressor will search through the listed names in order, in seeking an appropriate resource. The keyword ANYONE can be used to match any linked tracking receiver.

- WITH-TRACKER-GROUP
  LETHAL-ENGAGE-START

```
WITH-TRACKER-GROUP
  long_range_tracker_rx REQ-TO-SUSTAIN-ENG
  sam_trk_rx            OPT-TO-SUSTAIN-ENG
  med_range_rx          OPT-TO-SUSTAIN-ENG
  short_range_rx        OPT-TO-SUSTAIN-ENG
END TRACKER-GROUP
```

WITH-TRACKER-GROUP may only be used in the LETHAL-ENGAGE-START procedure. It is used to define a group of linked sensor receivers which can be used to track a target. All the named tracking sensor receivers must be present and available for use, i.e. meet the criteria outlined for the WITH-TRACKER qualifier, in order to commence the engagement. Those trackers labelled with the option REQ-TO-SUSTAIN-ENG are required for the engagement to continue, and at least one such tracking receiver must be present in the group. Those sensor receivers labelled with OPT-TO-SUSTAIN-ENG may be lost without the engagement being curtailed. This qualifier is useful to define a situation where one or more extra radars are used to illuminate a target to improve its visibility.

Each tracker may be turned on subsequent to the start of firing for use as a fire-control radar. This is accomplished with the TURN-ON option, for example:

```
WITH-TRACKER-GROUP
  short_range_tracker_rx REQ-TO-SUSTAIN-ENG
  high_power_rx          OPT-TO-SUSTAIN-ENG
    TURN-ON 0.5 (SEC) AFTER-FIRING
END TRACKER-GROUP
```

In the absence of a TURN-ON phrase all radars will be turned on at the commencement of the engagement; this precedes any decision to fire.

- WITH-ORDNANCE
  LETHAL-ENGAGE-FIRING-START

  ```
  WITH-ORDNANCE bullet
  ```

  WITH-ORDNANCE is required for use in the LETHAL-ENGAGE-FIRING-START procedure. It is used to specify the type of ordnance the allocated weapon system should use. More than one WITH-ORDNANCE entry can be given. The identified ordnance must either be named in the UAN as ORDNANCE or as a FUTURE-PLAYER.
- WITH-ELEMENT
  LETHAL-ENGAGE-FIRING-START

  ```
  WITH-ELEMENT sam_battery
  ```

  WITH-ELEMENT is required for use in the LETHAL-ENGAGE-FIRING-START procedure. It identifies the target element to be attacked with the allocated weapon. More than one potential target element can be identified, and as with the other options listed above, Suppressor will search for the first that is suitable. The keyword ANYONE can be used to allow any target to be selected. If this keyword is the last of several WITH-ELEMENT entries it will function as a catch-all default entry. (This is also the case with the WITH-TRACKER option.)
- WITH-SALVO-SIZE
  LETHAL-ENGAGE-FIRING-START

  ```
  WITH-SALVO-SIZE 6
  ```

  WITH-SALVO-SIZE can be used in the LETHAL-ENGAGE-FIRING-START procedure. It identifies the size of the salvo which is to be fired by the allocated weapon at the target. This item supercedes the obsolete TACTIC block entry SALVO-FIRING, which although still a legal Suppressor command, its use is not advised. Therefore the SALVO-FIRING command is not discussed in this guide.[19]
- WITH-PLAN
  LETHAL-ENGAGE-FIRING-START
  LAUNCH-START

  ```
  WITH-PLAN fly_out
  ```

  WITH-PLAN can be used in the LETHAL-ENGAGE-FIRING-START procedure and is required in the LAUNCH-START procedure. With the LAUNCH-START tactics this qualifier determines the name of the movement plan from the player's MOVE-PLANS followed by the name of the launched vehicle. If the plan specified requires

[19] Two other commands may be found in the Suppressor documentation which are either obsolete or not implemented, these are the TACTIC block keywords CENTRALIZATION-THRESHOLDS and NOMINAL-SUB-REACT-TIME. As with SALVO-FIRING these are not discussed in this guide.

arguments these are provided by the first mention of the plan in either the SDB `PLANS-FOR-MOVEMENT` or `PATH` statements for the player.

When used with `LETHAL-ENGAGE-FIRING-START` tactics `WITH-PLAN` describes the movement plan to be initially followed by the name of a `FUTURE-PLAYER` which is used as ordnance by the allocated weapon.

- `WITH-VEHICLE`
  `LAUNCH-START`

```
WITH-VEHICLE interceptor
```

`WITH-VEHICLE` is required in the `LAUNCH-START` procedure. It gives the vehicles, with either player names or future-player names, which can be launched by the tactical procedure. Several `WITH-VEHICLE` qualifiers can be given and Suppressor will search for the first available vehicle to launch. The name of the vehicle does not necessarily match the name of the resource in the `LAUNCH-START` procedure, since the resource is a subordinate which may in turn cause one of its resources to be launched.

- `WITH-TGT-CUING-FOR-LOC`
  `LETHAL-ASSIGNMENT`
  `LETHAL-ENGAGE`
  `GUNS-FREE/GUNS-TIGHT`
  `EMCON/ON-OFF`

```
WITH-TGT-CUING-FOR-LOC 4
```

`WITH-TGT-CUING` can be used with all lethal engagement, lethal assignment, lethal mode of control and emission control procedures. It sets the current heading of the specified locations of the resource to point at the target.

- `WITH-SDB-CUING-FOR-LOC`
  `LETHAL-ASSIGNMENT`
  `LETHAL-ENGAGE`
  `GUNS-FREE/GUNS-TIGHT`
  `EMCON/ON-OFF`

```
WITH-SDB-CUING-FOR-LOC 1
```

`WITH-SDB-CUING` can be used with all lethal engagement, lethal assignment, lethal mode of control and emission control procedures. It sets the current heading of the specified locations of the resource to that given by the appropriate SDB `HDG:` item. Neither `WITH-SDB-CUING-FOR-LOC` nor `WITH-TGT-CUING-FOR-LOC` should be used with moving players whose headings are aligned with their velocity vectors.

Apart from imposing any appropriate qualifying constraints on the selected resources the amount or number of the resources chosen must also be determined in the CHOOSE-FROM command. The PICK-AT-MOST keyword is used to describe this quantity:

```
PICK-AT-MOST 2 NOW
```

The above example would limit the current pipe to allocating or de-allocating no more than two subordinates. In other words if three fighter subordinates were available for deployment two of them would be allocated. If six fighters were active two would be de-allocated.

For procedures which allocate resources, (as opposed to those which de-allocate them), limits can be made on the total number allocated. The first limit that can be imposed is on the total number allocated by the player, for example we have:

```
PICK-AT-MOST 2 NOW 6 TOTAL
```

Here the keyword TOTAL limits the total allocation of the 'fighter_player' resource to six. Alternatively, or indeed additionally, a limit can be made on the total number of resources allocated by the current pipe; this uses the FILTER-TOTAL keyword, for example:

```
PICK-AT-MOST 2 NOW 6 TOTAL 4 FILTER-TOTAL
```

The example limits the total number of subordinates that may be allocated by this player to six with no more than four in total being selected by this pipe.

Finally each procedure ends with a simple one line trailer which indicates that the procedure is complete. In our example it is:

```
END LETHAL-ASSIGNMENT-QUEUE-ADD
```

Having completed the description of the syntax of a RESOURCE-ALLOCATION procedure the next stage in the process is to describe the function of each of the procedures. This is accomplished below, with each procedure's role being described in turn.

Lethal Assignment

- Description: Lethal assignment tactics control whether or not subordinates are assigned to attack targets perceived by their commander. Decisions are made on a target by target basis.
- Requirements: ASG-EVAL-RATE is a required entry of EVALUATION-RATES, it is used to specify the rate at which lethal assignment decisions will be made. ASG-TGT-UPDATE-RATE is an optional entry which may be used to confirm the lethal assignments at the specified rate.

Thinking Time	RESOURCE-ALLOCATION procedure
EVAL-ASSIGN-THREAT	LETHAL-ASSIGNMENT-QUEUE-ADD LETHAL-ASSIGNMENT-QUEUE-DROP
CONSIDER-ASG/CANCEL	LETHAL-ASSIGNMENT-START LETHAL-ASSIGNMENT-STOP

The four procedures controlling lethal assignment are organized into two pairs. The rate at which these decisions are made is determined by the occurrence of sensing chances and the ASG-EVAL-RATE and ASG-TGT-EVAL-RATE entries.

The first pair of procedures, LETHAL-ASSIGNMENT-QUEUE-ADD and LETHAL-ASSIGNMENT-QUEUE-DROP, determine which subordinates are currently candidates for a subsequent lethal assignment. These are evaluated at a rate controlled by sensing chances and the ASG-EVAL-RATE and require at least one thinker with the EVAL-ASSIGN-THREAT thinking time set in its TIME-TO-THINK capabilities.

The second pair of procedures, LETHAL-ASSIGNMENT-START and LETHAL-ASSIGNMENT-STOP, determine the conditions which cause assignments to be made or cancelled against these targets. The commander's thinkers will take a time determined by the CONSIDER-ASG/CANCEL thinking time to make these decisions and at least one thinker must be present with this thinking time set.

Lethal Engagement

- Description: Lethal engagement tactics control whether or not weapons are allocated to attack perceived targets. Decisions are made on a target by target basis. Requirements: ENG-EVAL-RATE is a required entry of EVALUATION-RATES, it gives the rate at which the commencement or cancelling of lethal engagements will be made.

Thinking Time	RESOURCE-ALLOCATION procedure
EVAL-ENGAGE-THREAT	LETHAL-ENGAGE-QUEUE-ADD LETHAL-ENGAGE-QUEUE-DROP
EVAL-LETHAL-ENGAGE	LETHAL-ENGAGE-START LETHAL-ENGAGE-STOP
EVAL-FIRING	LETHAL-ENGAGE-FIRING-START LETHAL-ENGAGE-FIRING-STOP

The six procedures controlling lethal engagement are organized into three pairs. The rate at which these decisions are made is determined by the occurrence of sensing chances and the ENG-EVAL-RATE entry.

The first pair of procedures, LETHAL-ENGAGE-QUEUE-ADD and LETHAL-ENGAGE-QUEUE-DROP, determine which players are currently candidates for a subsequent lethal engagement. At least one thinker with the EVAL-ENGAGE-THREAT thinking time set in its TIME-TO-THINK capabilities is required.

The second pair of procedures, LETHAL-ENGAGE-START and LETHAL-ENGAGE-STOP determine the conditions which allow engagements to commence and cause them to cease against these targets. The commander's thinkers will take a time determined by the EVAL-LETHAL-ENGAGE thinking time to make these decisions. At least one thinker must be present with this thinking time set.

The third and final pair of procedures, LETHAL-ENGAGE-FIRING-START and LETHAL-ENGAGE-FIRING-STOP determine when a weapon will open and cease fire against the selected targets. Thinkers with the EVAL-FIRING thinking time are required for these procedures.

Non-Lethal Engagement

- Description: Non-lethal engagement tactics control whether or not disruptors are allocated to disrupt detected systems. Decisions are made on a target by target basis.
- Requirements: `JAM-EVAL-RATE` is a required entry of `EVALUATION-RATES`, it gives the rate at which the commencement or cancelling of non-lethal engagements will be made.

Thinking Time	RESOURCE-ALLOCATION procedure
EVAL-JMR-QUEUE	JAMMER-QUEUE-ADD JAMMER-QUEUE-DROP
EVAL-JMR-SPOTS	JAMMER-SPOT-ADD JAMMER-SPOT-DROP

The four procedures controlling non-lethal engagement are organized into two pairs. The rate at which these decisions are made is determined by the occurrence of sensing chances and the `JAM-EVAL-RATE` entry.

The first pair of procedures, `JAMMER-QUEUE-ADD` and `JAMMER-QUEUE-DROP`, determine which systems are currently candidates for jamming. One or more thinkers with the `EVAL-JMR-QUEUE` thinking time must be present.

The second pair of procedures, `JAMMER-SPOT-ADD` and `JAMMER-SPOT-DROP`, control when and how jammers should actively jam a target system. The thinkers evaluating these procedures must have the `EVAL-JMR-SPOTS` time-to-think entry defined.

Launch Movement

- Description: Launch movement tactics control the launching of a resource into motion. Decisions are made globally.
- Requirements: `LAUNCH-EVAL-RATE` is a required entry of `EVALUATION-RATES`, it gives the rate at which the conditions to launch resources will be evaluated.

Thinking Time	RESOURCE-ALLOCATION procedure
CONSIDER-LAUNCH	LAUNCH-START

There is a single procedure which decides whether or not to launch a resource. Clearly the action of launching cannot vary from target to target and so is not considered separately for each target. Furthermore, once a resource is launched the action cannot be revoked, so there is no concept of 'launch-stop'. At least one thinker with the `CONSIDER-LAUNCH` thinking time must be present.

Emission Control

- Description: Emission control tactics control the turning on and off of a player's sensors. Decisions are made globally.
- Requirements: `EMCON-EVAL-RATE` is a required entry of `EVALUATION-RATES`, it gives the rate at which the conditions to turn on and off sensors will be evaluated.

Thinking Time	RESOURCE-ALLOCATION procedure
EVAL-EMCON-CHANGE	EMCON/TURN-ON EMCON/TURN-OFF

A single pair of procedures are used for emission control, `EMCON/TURN-ON` and `EMCON/TURN-OFF`. The player will think about these options at a rate specified by the `EMCON-EVAL-RATE` and a thinker with the `EVAL-EMCON-CHANGE` thinking time must be available.

The ability of a player to consider these tactics depends on the value of the SDB `MODES-OF-CONTROL:` entry `EMCON`. If this is set to be `CMDR` or some other player than the current player, then these tactics are only considered whilst the player has active target assignments from a commander. If the player acquires lethal mode of control or if the `EMCON` entry is `SELF` or the name of the current player then it will be able to make emission control evaluations independently of receiving assignments.

Change Lethal Mode of Control

- Description: Lethal mode of control tactics control whether or not the current player's subordinates have the authority to initiate lethal engagements, 'guns free', or must wait for assignments from commanders, 'guns tight'. Decisions are made on a subordinate by subordinate basis.
- Requirements: `FREE/TIGHT-EVAL-RATE` is a required entry of `EVALUATION-RATES`, it gives the rate at which the settings of the subordinates' lethal modes of control are considered.

Thinking Time	RESOURCE-ALLOCATION procedure
EVAL-GUNS-FREE/TIGHT	GUNS-FREE GUNS-TIGHT

One pair of procedures, `GUNS-FREE` and `GUNS-TIGHT`, determine when 'guns free' or 'guns tight' orders are sent to each subordinate. The former order gives the subordinate authority to initiate lethal engagements, the latter constrains the subordinate to await a lethal assignment order. The player evaluating the tactics must have at least one thinker with the `EVAL-GUNS-FREE/TIGHT` thinking time set.

3.6.2.1 A Resource Allocation Example

A complete example of a RESOURCE-ALLOCATION block for lethal engagement can now be given. The tactics illustrated are quite a bit simpler than those that might be required for realistic scenarios but they give a flavour of what is required in preparing a Suppressor data base. Obvious simplifications are, that the fighter can consider any target and has but a single weapon system, a missile launcher.

Lethal Engage Queue Add and Drop

```
$ Lethal Engagement Procedures for a simple fighter model
LETHAL-ENGAGE-QUEUE-ADD
  TGT-TYPE ANYONE
    USE INPUT FOR FILTER 1
      WPN-TYPE   missile_launcher
        IFF-STATUS IS-NOT FRIEND
          AND 3D-TGT-LOC WITHIN aew_command_zone
          AND WPN-STATUS IS-NOT WPN-NON/OP
          AND BELIEVED-ALIVE
    USE FILTER 1 SELECTIONS FOR FILTER 2
      WPN-TYPE   missile_launcher
        BEEN-ASSIGNED IS YES
          OR ENG-CONTROL-MODE IS SELF
    FROM FILTER 2 SELECTIONS
      CHOOSE-FROM missile_launcher
      PICK-AT-MOST 1 NOW 1 TOTAL
END LETHAL-ENGAGE-QUEUE-ADD
```

The first filter initially checks to see if the candidate target is a friend or foe. This is accomplished with the IFF-STATUS condition:

```
IFF-STATUS IS-NOT FRIEND
```

Apart from the rejection of friendly players as targets the first filter checks certain other conditions. The first is that the target is inside the zone 'aew_command_zone', whose location is defined within the SDB. The second confirms that the weapon specified as the current resource, i.e. 'missile_launcher', is functioning. Finally, the player checks to see if the target is alive or dead. When all these conditions are satisfied the first filter is complete and any targets that meet all the criteria are passed as output from the filter.

In our example the output of the first filter goes solely to the second filter which, in this example is simply:

```
USE FILTER 1 SELECTIONS FOR FILTER 2
  WPN-TYPE   missile_launcher
    BEEN-ASSIGNED IS YES
      OR ENG-CONTROL-MODE IS SELF
```

This checks that either the player has been assigned by a commander to engage the target or it has the authority to initiate its own engagements. This authority is conferred by the 'MODES-OF-CONTROL:' item in the SDB.

The procedure which removes targets from the lethal engagement queue for the fighter player is very similar in form to that which adds the targets to the queue:

```
LETHAL-ENGAGE-QUEUE-DROP
  TGT-TYPE ANYONE
    USE INPUT FOR FILTER 1
      WPN-TYPE missile_launcher
        IFF-STATUS IS FRIEND
          OR 3D-TGT-LOC OUTSIDE aew_command_zone
             AND ENG-CONTROL-MODE IS SELF
          OR BEEN-ASSIGNED IS NO
             AND ENG-CONTROL-MODE IS-NOT SELF
          OR WPN-STATUS IS WPN-NON/OP
          OR BELIEVED-DEAD
          OR AVAILABLE-RESOURCE NO-MORE-THAN 0 (ROUNDS)
             RE: missile ORDNANCE
          OR SALVOS-FIRED-DURING-ENG AT-LEAST 2 (SALVOS)
             RE: missile ORDNANCE
    FROM FILTER 1 SELECTIONS
      CHOOSE-FROM missile_launcher
      PICK-AT-MOST 1 NOW
END LETHAL-ENGAGE-QUEUE-DROP
```

We notice here that just one filter is used to deallocate the resource in this procedure. One noteworthy feature of this procedure is that both AND and OR conductions are used in the filter. As remarked earlier, AND statements take precedence over OR statements, and the indentation is intended to serve as a visual reminder of this.

The conditions that cause a target to be de-allocated in this example are the following:

1. the target is perceived to be friendly,
2. the target is outside the command zone when the player has authority to initiate engagements,
3. the target is no longer assigned when the player requires an assignment to engage the target,
4. the weapon is no longer operational,
5. the target is believed dead,
6. the weapon's ordnance is exhausted, i.e. its missiles are all expended, and
7. at least two salvos have already been fired at the target.

Once a target is entered upon the queue the next decision is whether or not to commence an engagement against it. The procedure `LETHAL-ENGAGE-START` contains the criteria for this decision. It is shown below for the fighter player-type and requires just one filter.

```
LETHAL-ENGAGE-START
  TGT-TYPE ANYONE
    USE INPUT FOR FILTER 1
      WPN-TYPE missile_launcher
       REL-TGT-ALT > -12.0 (KM) AND REL-TGT-ALT < 12.0 (KM)
        AND 2D-REL-TGT-OFFSET <     49.5   (KM)
        AND 2D-DIST           < 54995.0   (M)
        AND REL-TGT-HDG       <     99.01 (DEG)
        AND SALVOS-FIRED-DURING-ENG NO-MORE-THAN 2 (SALVOS)
          RE: missile ORDNANCE
    FROM FILTER 1 SELECTIONS
      CHOOSE-FROM missile_launcher
        WITH-TRACKER fighter_radar_rx
      PICK-AT-MOST 1 NOW 1 TOTAL
END LETHAL-ENGAGE-START
```

We note here that several of the conditions now test continuous ranges, so that, for example, the statement:

```
REL-TGT-ALT > -12.0 (KM) AND REL-TGT-ALT < 12.0 (KM)
```

ensures that the relative altitude difference between the player's resource and the target is within the range from 12 km to -12 km. In fact most of the conditions are on the relative positions and headings of the resource and the target. Finally, the `CHOOSE-FROM` keyword is used to select the weapon, which here requires that the tracking radar 'fighter_radar_rx' is operational.

A very similar procedure may be used to stop an engagement once it has commenced. Again a single filter suffices which deallocates the resource when the target is too far away or insufficient ordnance remains. The choice of thresholds for the criteria is such that those which allow engagements to start are slightly more relaxed than those which are used to cancel an ongoing engagement. This overlap prevents a marginal decision to start an engagement being immediately cancelled, and this cancellation immediately being overridden and so on.

```
LETHAL-ENGAGE-STOP
  TGT-TYPE ANYONE
    USE INPUT FOR FILTER 1
      WPN-TYPE missile_launcher
        REL-TGT-ALT < -13.0 (KM) OR REL-TGT-ALT > 13.0 (KM)
          OR 2D-REL-TGT-OFFSET >    51.75 (KM)
          OR 2D-DIST            > 57500.00  (M)
          OR REL-TGT-HDG        >   103.51 (DEG)
          OR AVAILABLE-RESOURCE NO-MORE-THAN 0 (ROUNDS)
            RE: missile ORDNANCE
          OR SALVOS-FIRED-DURING-ENG AT-LEAST 2 (SALVOS)
            RE: missile ORDNANCE
    FROM FILTER 1 SELECTIONS
      CHOOSE-FROM missile_launcher
      PICK-AT-MOST 1 NOW
END LETHAL-ENGAGE-STOP
```

In the lethal engagement procedural grouping the final two procedures define the criteria for firing a weapon at a target. The procedure to open fire is relatively complicated, introducing multiple target paragraphs for the first time. The first paragraph deals with targets of the type 'bomber_player', and uses two filters. The first filter makes a preliminary selection according to the relative speed and position of the target:

```
LETHAL-ENGAGE-FIRING-START
TGT-TYPE bomber_player
  USE INPUT FOR FILTER 1
    WPN-TYPE missile_launcher
      REL-TGT-ALT > -7.0 (KM) AND REL-TGT-ALT < 7.0 (KM)
        AND 2D-REL-TGT-OFFSET < 38.25 (KM)
        AND 2D-DIST > 3542.5 (M) AND 2D-DIST < 42507.5 (M)
        AND REL-TGT-HDG < 76.51 (DEG)
        AND FIRING-NOW IS NO
        AND TRACKING-STATUS IS TRACKING
  USE FILTER 1 SELECTIONS FOR FILTER 2
    WPN-TYPE missile_launcher
      REL-TGT-ALT > -6.0 (KM) AND REL-TGT-ALT < 6.0 (KM)
        AND 2D-REL-TGT-OFFSET <    36.0   (KM)
        AND 2D-DIST           < 40010.0   (M)
        AND REL-TGT-HDG       <    72.01  (DEG)
        AND SALVOS-FIRED-DURING-ENG NO-MORE-THAN 1 (SALVOS)
          RE: missile ORDNANCE
  FROM FILTER 2 SELECTIONS
    CHOOSE-FROM missile_launcher
      WITH-ORDNANCE missile
      WITH-ELEMENT ANYONE
      WITH-SALVO-SIZE 2
    PICK-AT-MOST 1 NOW 1 TOTAL
```

Firstly, note that the conditions on the speed and the position must also satisfy those of the `LETHAL-ENGAGE-START` otherwise the firing can never commence. In addition to these constraints a check is made that the weapon is not already in use and that the weapon's tracking radar, ('fighter_radar_rx'), is actively tracking the target. When these conditions are all met the output is passed to the second filter which is the last in this pipe.

A slightly tighter range of position and motion constraints are set before firing can commence and the thinker checks that no more that one salvo was previously fired at the target. If the conditions in this filter fail the thinker now checks a third filter, which represents a second branch of a forked pipe:

```
USE FILTER 1 SELECTIONS FOR FILTER 3
  WPN-TYPE missile_launcher
    REL-TGT-ALT > -6.0 (KM) AND REL-TGT-ALT < 6.0 (KM)
      AND 2D-REL-TGT-OFFSET <     36.0  (KM)
      AND 2D-DIST           < 40010.0   (M)
      AND REL-TGT-HDG       <     72.01 (DEG)
      AND SALVOS-FIRED-DURING-ENG NO-MORE-THAN 2 (SALVOS)
        RE: missile ORDNANCE
FROM FILTER 3 SELECTIONS
  CHOOSE-FROM missile_launcher
    WITH-ORDNANCE missile
    WITH-ELEMENT ANYONE
    WITH-SALVO-SIZE 1
  PICK-AT-MOST 1 NOW 1 TOTAL
```

Close inspection reveals that the third filter differs from the second filter only in the criterion:

```
AND SALVOS-FIRED-DURING-ENG NO-MORE-THAN 2 (SALVOS)
```

and in the `WITH-SALVO-SIZE` qualifier at the end of this pipe:

```
WITH-SALVO-SIZE 1
```

So if the weapon is allocated by the first pipe then a salvo size of two will be used by the weapon, otherwise a salvo size of one will be used. The forked pipe is a convenient way of producing this dynamic change in the size of the fired salvo.

The second target paragraph selects any targets other than the bomber player-type and uses the same form for the filters but now always fires a salvo with just a single missile at a selected target.

```
TGT-TYPE ALL-OTHERS
  USE INPUT FOR FILTER 1
    WPN-TYPE missile_launcher
      REL-TGT-ALT > -7.0 (KM) AND REL-TGT-ALT < 7.0 (KM)
       AND 2D-REL-TGT-OFFSET < 38.25 (KM)
       AND 2D-DIST > 3542.5 (M) AND 2D-DIST < 42507.5 (M)
       AND REL-TGT-HDG < 76.51 (DEG)
       AND FIRING-NOW IS NO AND TRACKING-STATUS IS TRACKING
  USE FILTER 1 SELECTIONS FOR FILTER 2
    WPN-TYPE missile_launcher
      REL-TGT-ALT > -6.0 (KM) AND REL-TGT-ALT < 6.0 (KM)
       AND 2D-REL-TGT-OFFSET <     36.0  (KM)
       AND 2D-DIST           <  40010.0  (M)
       AND REL-TGT-HDG       <     72.01 (DEG)
  FROM FILTER 2 SELECTIONS
    CHOOSE-FROM missile_launcher
      WITH-ORDNANCE missile
      WITH-ELEMENT ANYONE
    PICK-AT-MOST 1 NOW 1 TOTAL
```

In comparison to the procedure to commence firing the tactics which control the ending of a firing sequence are quite simple. The procedure employs a single filter which ceases fire when either the target has moved out of range or is believed to have been destroyed:

```
LETHAL-ENGAGE-FIRING-STOP
  TGT-TYPE ANYONE
    USE INPUT FOR FILTER 1
      WPN-TYPE missile_launcher
        REL-TGT-ALT < -9.0 (KM) OR REL-TGT-ALT > 9.0 (KM)
          OR 2D-REL-TGT-OFFSET > 42.5 (KM)
          OR 2D-DIST < 2547.5 (M) OR 2D-DIST > 47502.5 (M)
          OR REL-TGT-HDG > 85.51 (DEG)
          OR BELIEVED-DEAD
    FROM FILTER 1 SELECTIONS
      CHOOSE-FROM missile_launcher
      PICK-AT-MOST 1 NOW
END LETHAL-ENGAGE-FIRING-STOP
```

Notice that missiles will not be fired when the target is too close as well as too far away; this prevents the player from damaging itself in the engagement.

The discussion of the `RESOURCE-ALLOCATION` reactive tactics is now complete, but it remains to describe the `MOVE-PLANS` tactics. These also describe reactive thought processes, namely those which a player uses to manoeuvre during a Suppressor scenario.

3.6.3 The Tactics of Movement

The TDB instructions dealing with movement are particularly important and relatively complicated. Suppressor divides movements into two classes, reactive manoeuvres and pre-programmed manoeuvres. Reactive manoeuvres may themselves be subdivided into manoeuvres designed to attack a target, and those to avoid a threat or terrain. The manoeuvres performed in approaching a target are in fact sets of detailed instructions describing where the attacking mover should be relative to the position of the target. These differ from pre-programmed manoeuvres only because the target is at an initially undetermined location and may itself be moving. Terrain and threat avoidance is achieved by Suppressor changing pre-programmed paths to avoid perceived threats and the terrain. Suppressor will not change reactive manoeuvres to avoid the ground or a threat; the reactive move plans must themselves deal with these eventualities.

Manoeuvring Against Targets

Players that can reactively move against targets must possess some means of perceiving these targets. This would mean a sensor system (for direct perception) or a communication receiver (for indirect perception). Furthermore, permission to move to engage a target must be granted by the command:

```
MOVE-TO-ENG YES
```

The default being that no permission is granted:

```
MOVE-TO-ENG NO
```

The targets that may be engaged are identified with the `ATK-PRIORITIES` command:

```
ATK-PRIORITIES
  DIMENSION 1 LIST-NAME air_targets
    TGT-ELEMENTS bomber fighter/bomber
END ATK-PRIORITIES
```

Here we have allocated two types of *elements* as being eligible for targetting. (Notice that elements and not player-types are targets, this is because only one physical location may be targeted at once and a player may be distributed over several locations.) There is no limit to the number of elements that can be allocated to each target class and each element may be allocated to several different classes. In fact there are no restrictions on these allocations at all. The name of the list, in this example 'air_targets', must be included in the UAN section as a `MANEUVER`.

The ATK-PRIORITIES entry basically labels the potential targets of a player in a way in which the MOVE-PLANS can later understand. The actual selection of these targets is made with the FOCUS-ON-PRIORITY entry of a MOVE-PLANS tactical block itself. For example:

```
FOCUS-ON PRIORITY air_targets
```

would select targets of type bomber and fighter/bomber as candidate targets. Since a mover can only approach one target at a time the first such target that meets the requirements of the MOVE-PLANS tactics will be selected as the current target. The mover will then start to reactively manoeuvre to engage this target. Great care should be made that the tactics expressed in the lethal engagement blocks are consistent with this choice, since Suppressor cannot do this automatically.

When a mover manoeuvres to attack a target it needs to compute its route in terms of the target's current position. When the target is moving this route will need constant refinement. To minimize the changes in direction the mover must take, the mover may choose to close not on the target's current position but on the predicted intercept position. This is the default behaviour of Suppressor and is equivalent to using the INTERCEPT-MODE instruction to set:

```
INTERCEPT-MODE PREDICTED
```

If the target maintains a constant speed and heading the mover will not need to change its course.

Alternatively, it may be preferred that the mover approach the target itself since this will naturally cause the mover to tuck in behind a moving target and approach it from the rear. This may be set with the instruction:

```
INTERCEPT-MODE PURSUIT
```

When pursuit mode is selected a further refinement is possible: a leading or lagging offset may be set so that the attacker closes on a point some way fore or aft of the target. Hence the instruction:

```
PURSUIT-MODE-OFFSET   1.0 (KM)
```

sets an intercept point 1 km in front of the target, whilst

```
PURSUIT-MODE-OFFSET  -0.5 (KM)
```

sets an offset 0.5 km behind the target.

The reactive behaviour of the attacker as it pursues a target may also be modified with the `REVECTOR-DIST-THRESH` instruction. Normally a player will react to any change in the intercept point and change its course immediately. If the ratio of the change in the intercept point's position to the current separation is small this would produce many small and unnecessary course changes. If instead a set of thresholds are defined specifying the minimum change in the intercept point required to produce a course change, these superfluous manoeuvres may be eliminated.

```
$ Example distance threshold table
REVECTOR-DIST-THRESH
  DIMENSION 1 2D-DIST-REL-INT (KM)
      0.0 10.0 20.0 40.0 80.0 160.0 1000.0
    INTERCEPT-CHANGE (KM)
        0.0  0.5  1.0  2.0  4.0  10.0
END REVECTOR-DIST-THRESH
```

The above table defines six distance intervals covering a range of 1000 km and so has six intercept thresholds. Below a 10 km range, all changes in the intercept position are responded to; from 10 km to 160 km all changes of more than 0.05 radians are tracked; beyond that the computed intercept point must change by at least 10 km to provoke a response. The actual values used would depend very strongly on the physical capabilities (speed, acceleration and turn-radius) of the players involved.

The nature of a player's path in space may also be modified with the `ACCELERATION-MODE` instruction. Normally, Suppressor uses uniform acceleration in varying a player's speed when moving from one point on its path to another. This means that the mover will accelerate at a constant rate whilst travelling from one checkpoint to the next. With this pattern of acceleration, note that the mover will always achieve the new speed regardless of the mover's physical capabilities. (So that the `MAX-ACCELERATION` setting of the capability block is ignored.) The default is equivalent to setting:

```
ACCELERATION-MODE  UNIFORM
```

whereas an alternative is to set:

```
ACCELERATION-MODE  MAXIMUM
```

This has the effect of always changing the player's speed using the maximum value of its acceleration, (as defined in the `CAPABILITY` block using `MAX-ACCELERATION`). If the desired speed cannot be reached before the next way-point on the path is reached the speed actually achieved will be used, and Suppressor will print a message to the listing file informing the user of the alteration in the speed from that specified in the SDB.

Terrain Following, Terrain Avoidance and Threat Avoidance

Terrain following and terrain and threat avoidance are conferred with the MOVE-OPTIONS instruction. For example the instruction:

```
$ Only enable terrain following
MOVE-OPTIONS
  TERRAIN-FOLLOW
END MOVE-OPTIONS
```

would enable a player to follow terrain by changing its altitude to maintain a fixed height above ground level. This height is initially specified in the BOUNDARY instruction in the SDB. More than one option may be set at once, as with

```
$ Enable both terrain and threat avoidance
MOVE-OPTIONS
  TERRAIN-AVOID
  THREAT-AVOID
END MOVE-OPTIONS
```

which simultaneously enables threat avoidance and terrain avoidance, (where obstacles such as mountains may be flown around whilst maintaining the altitude within some fixed bounds), but not terrain following. If the MOVE-OPTION instruction is not used then this is the same as setting:

```
$ Disable everything (the default)
MOVE-OPTIONS
  NONE
END MOVE-OPTIONS
```

where none of the above features are enabled. All three features may be enabled together or in any combination.

If terrain following is enabled then a LOOK-AHEAD-DISTANCE must also be specified which says how far ahead a player may look in order to decide whether to climb or dive to follow the terrain. The greater this distance the smoother the path flown. Its format is of the form:

```
LOOK-AHEAD-DISTANCE 25.0 (KM)
```

Threat avoidance is implemented in Suppressor by defining, for each player-type with the 'threat-avoidance' capability, a set of 'threat-volumes' around each player-type that potentially poses a threat. The format of the threat-volume instruction is as shown in the following example:

```
$ This player type will try to avoid sam sites
$ First enable the threat avoidance capability
MOVE-OPTIONS THREAT-AVOIDANCE END MOVE-OPTIONS
$ Now define the player-types to be avoided and assign
$ priorities to different volumes around the type
THREAT-VOLUME
  DIMENSION 1  PLAYER-TYPE  sam_player
    DIMENSION 2  ALT (M)    -20.0  500.0  10.0E3
      DIMENSION 3  RNG (KM)  0.0    1.0    3.0   10.0
        PRIORITY (NO-UNITS)  5      4      1
      DIMENSION 3  RNG (KM)  0.0    75.0 100.0
        PRIORITY (NO-UNITS)  5      4
END THREAT-VOLUME
```

The first entry of the table lists the player-types to be avoided (just one entry here). The next entry lists a range of altitudes and then a third list gives the various ranges for each altitude pair. Finally, a priority is assigned for each radius-altitude range combination: the higher the priority the more keenly the mover will feel the desire to avoid the area. For the example shown, the player-type will be prepared to approach much closer to the SAM site at low altitude (less than 500.0 m above the SAM site), than at high altitudes. The threat avoidance is defined in terms of player-types in the TDB and the list of actual players that are to be avoided is given in a `TOLD ABOUT` statement in the SDB instruction set. (Some of the friendly players in a model may be of the potentially threatening type and if the probability of fratricide is low it may be acceptable not to avoid these players.) Note that since the threat avoidance is around a player all the (physical) locations of that player will be avoided. For example, the 'sam_player' may have multiple missile batteries at different physical locations; if so each should be avoided.

It may be the case that a player cannot avoid entering a threat-volume; this would occur when all routes that a player may take in reaching its goal would enter at least one dangerous zone. In this case Suppressor will first seek a route that enters zones of lowest priority only, continuing in this manner until a route is found, even if this should mean ignoring the threat-volume completely.

Move Plans

The movement plans used by a player that may reactively manoeuvre are given in the `MOVE-PLANS` data block amongst the player-type's tactics. Just like `RESOURCE-ALLOCATION` procedures these are tactics in the form of simple procedures which are processed by thinkers. Unlike `RESOURCE-ALLOCATION` procedures there are no user controlled evaluation rates, but a thinking time `CONSIDER-MOVE` must be set for at least one thinker with the moving player. The table first introduced for the thinker's capability block can be reproduced here:

Thinking Time	Associate Tactics
CONSIDER-MOVE	MOVE-PLANS

The `MOVE-PLANS` block also acts as a place-holder for systems without reactive movement capabilities. The simplest case would be a player with an entire sequence of movements specified by the SDB. Here an empty `MOVE-PLANS` block should be given:

```
MOVE-PLANS
END MOVE-PLANS
```

The AEW player-type introduced earlier has entirely pre-programmed manoeuvres, but they include an important type of manoeuvre that is defined within the TDB, namely the `PATTERN` manoeuvre:

```
MOVE-PLANS
  PLAN begin_path
  END-PLAN
  PLAN start_orbit
    NOW-USE PATTERN racetrack
  END-PLAN
END MOVE-PLANS
```

This `MOVE-PLANS` block declares that two named plans ('begin_path' and 'start_orbit') are to be followed by the AEW player-type. The first of these is the simplest of all possible plans; it is just a named plan whose precise itinerary is given in full in the SDB. The second plan, 'start_orbit', is slightly more complex; it instructs an AEW player to follow a movement pattern named 'racetrack' once it has reached the starting point of the plan 'start_orbit'. The command which invokes the pattern:

```
NOW-USE PATTERN racetrack
```

is an example of an 'action' statement within a `MOVE-PLANS` block. These statements determine what course of action is taken by the player whilst following a particular plan. In this case, it invokes a pattern named 'racetrack', whose format will be described later along with the other legal action statements.

More complex MOVE-PLANS will also include conditional statements formed using the tactical criteria available to these plans. These were described in detail earlier and are very similar to those used for RESOURCE-ALLOCATION procedures.

Conditional Statements

The major difference between the structure of a RESOURCE-ALLOCATION condition and a MOVE-PLANS condition is the existence of a WHEN statement within the MOVE-PLANS tactics. This is composed of three distinct statements, namely WHEN, BUT-WHEN and OTHERWISE. Their function may be summarized by a single example:

```
WHEN {Condition A}
  <An action that occurs when A is true>
BUT-WHEN {Condition B}
  <An action that occurs when A is false and B is true>
OTHERWISE
  <An action that occurs when both A and B are false>
```

A real PLAN need not have exactly one of each of these items. If any at all are used then an initial WHEN clause will be required for each conditional block. Although blocks may be nested they may not be concatenated within a single plan, but this effect may be achieved by joining together different plans in sequence. Subsequent to the initial WHEN clause any number of BUT-WHEN clauses may be used, including none at all. Exactly one OTHERWISE clause is always required to determine what to do should none of the specified conditions be currently true. It should be noted that despite the nomenclature of 'when ... but-when' the first true condition is enacted, as in an 'if ... else' construct in many programming languages.

The real complexity with conditional statements lies not with the overall structure of the WHEN statement but in the various criteria that may be used within each condition. These have already been discussed in detail and an example is:

```
WHEN PERCEPTION-SOURCE IS DIRECT-INTELL
```

which ensures that the source of a target perception is due to a player's own sensor and not from a received message. This is about as simple as a condition can be, and Suppressor allows a single condition to be made up of one or more criteria joined together with the logical operator AND, for example:

```
WHEN SNR-STATUS IS LOSE-DETECT
     AND REL-TGT-ALT < -600.0 M
```

would apply a condition that is true when the player has just ceased some process in the lethal engagement of a target and is simultaneously more than 600 m below the target.

The SNR-STATUS criterion is an important one since it provides a way to co-ordinate the activities of a mover with the targets selected in its lethal engagement tactics. Briefly SNR-STATUS IS DETECT will be true if:

- the current target of the MOVE-PLANS plan is a member of lethal engage queue; (i.e. the current target passes LETHAL-ENGAGE-QUEUE-ADD), or
- the current target is being lethally engaged; (i.e. the current target passes LETHAL-ENGAGE-START), or
- the current target is being fired at, (i.e. the current target passes LETHAL-ENGAGE-FIRING-START).

Conversely SNR-STATUS IS LOSE-DETECT will be true if:

- firing at the current target has just stopped; (i.e. the current target passes LETHAL-ENGAGE-FIRING-STOP), or
- lethal engagement of the current target has just stopped; (i.e. the current target passes LETHAL-ENGAGE-STOP), or
- the current target has been dropped from the lethal engagement queue, (i.e. the current target passes LETHAL-ENGAGE-QUEUE-DROP).

Notice that the correct use of SNR-STATUS requires that the mover has LETHAL-ENGAGE tactics defined for it, and that the target be a legitimate target both for reactive manoeuvring and for lethal engagement.

Unlike RESOURCE-ALLOCATION procedures the logical inclusive or operator 'OR' is not provided by MOVE-PLANS procedures. It can however be emulated by using the BUT-WHEN statement. Hence if we had wanted to say:

```
WHEN SNR-STATUS IS LOSE-DETECT
     OR REL-TGT-ALT < -600.0 M
  <execute actions>
```

we must instead say:

```
WHEN SNR-STATUS IS LOSE-DETECT
  <execute actions>
BUT-WHEN REL-TGT-ALT < -600.0 M
  <execute the same actions above>
```

Suppressor allows nesting of conditional statements; this can be accomplished with the `THEN:` and `END-THEN.` keywords. Note that he colon and full-stop are part of the keywords and are required. The following example nests a second `WHEN` statement within the first:

```
WHEN SNR-STATUS IS DETECT
  THEN:
    WHEN TGT-TYPE IS some_target
      <Action to take>
    OTHERWISE
      <Default action to take>
  END-THEN.
OTHERWISE
  FOCUS-ON PRIORITY priority
```

Nesting can be many levels deep, with a maximum depth subject only to the constraints of available machine memory.

Action Statements

The action statements describe precisely what action a mover is to take at any moment of the scenario. Which action statements are followed depends on the status of the conditional statements which bracket the statements themselves.

The fourteen possible action statements are:

- `NOW-USE PATTERN`
  This causes the mover to follow a 3D manoeuvre known as a `PATTERN` which may repeat and is defined in terms of the mover's location on commencement of the pattern. Each pattern may be used quite flexibly; firstly the whole manoeuvre may be rotated to be aligned with the mover's initial heading on commencing the pattern, for example:

```
PLAN start_orbit
  NOW-USE PATTERN racetrack USING REL-HEADING
END-PLAN
```

In addition, sensors may be turned on or off during phases of the planned manoeuvre, so that:

```
PLAN start_orbit
  NOW-USE PATTERN racetrack
  USING TURN OFF 1030 aewc_radar_rx DURING TURN
END-PLAN
```

would turn off the AEW player's sensor receiver during turns. Other phases that may be specified are CLIMB and DIVE and any alphabetically ordered combination of these, such as CLIMB/DIVE/TURN or DIVE/TURN.

- NOW-USE PROFILE
  A PROFILE is a vertical manoeuvre, i.e. it is defined in the vertical plane defined by the target's range vector. Unlike a PATTERN the manoeuvre is defined in terms of a target's position.
- NOW-USE ASPECT
  An ASPECT is a 3D manoeuvre defined in terms of the target's position. It allows more control of the position of the mover than a PROFILE.
- NOW-USE INTERCEPT-MODE
  This item allows the mover to change its current INTERCEPT-MODE during a movement plan. When PURSUIT mode is selected the PURSUIT-MODE-OFFSET may also be set dynamically, for example:

```
NOW-USE INTERCEPT-MODE PURSUIT WITH-OFFSET 1.0 (KM)
```

  would change the offset to be 1 km.
- EXECUTE PLAN
  A new plan may be executed as an action, and the current plan may be called recursively. Arguments may be passed to plans that expect them by including them within parentheses that are separated from surrounding words with white space. An example might be:

```
EXECUTE PLAN resume_patrol ( All_threats )
```

  where 'All_threats' is a dummy argument to be passed to the plan 'resume_patrol'. When a new plan is to be executed the contents of the new plan will not be evaluated and processed until the next time the thinker evaluates its movement plans.
- EXECUTE SUSPEND MOVEMENT
  A mover may be stopped with the expectation that it will later move again with this action statement. When conjoined with either GOTO instruction, or a PROFILE or ASPECT, instruction the mover will stop after the other instruction is completed. This action will have no effect when used with a PATTERN or a EXECUTE SDB-PLAN instruction.

  A mover will not stop instantly but must be decelerated at the rate specified by the MAX-ACCELERATION item of the mover's CAPABILITY block, which means it will stop at a later time and at a different place from where it was when the command was issued. Note that Suppressor is not familiar with the law of gravity and will allow a careless pilot to stop a fully laden tanker aircraft in mid-air.

  No explicit instruction exists to resume movement but various action statements will implicitly restart the mover. These instructions are any that involve motion, namely the PATTERN, PROFILE, ASPECT or either GOTO statement. A problem here is that the mover will need to achieve some speed which should be specified by the instruction. All the above instructions specify a speed for the mover except for GOTO POSITION TGT, so in this case the mover will just accelerate at its MAX-ACCELERATION-RATE until its maximum speed capability is reached. This may well not be a realistic speed so the GOTO POSITION TGT action statement should only be used with care in order to restart a still mover.

- GOTO POINT
  This simply causes the mover to go to the specified point. For example:

```
GOTO POINT charlie
```

would make the mover head towards the checkpoint 'charlie' defined in the mover's SDB entry.

- EXECUTE SDB-PLAN AT-CHECKPOINT
  This action must be used in conjunction with a GOTO POINT action statement with the result that the mover will resume its planned path once it has reached the indicated checkpoint. So, an example of its use might be:

```
GOTO POINT charlie
AND EXECUTE SDB-PLAN AT-CHECKPOINT
```

- GOTO POSITION TGT
  This action statement has one of two effects depending on context. If used with a PROFILE instruction then the mover will follow that PROFILE towards the current perceived position of the target. Used on its own the mover will move directly towards the target's current position, i.e. will move without the aid of a PROFILE. Note that the target itself might move away from this position before the mover arrives there.
- FOCUS-ON PRIORITY
  This entry informs the mover's thinker what targets it is to consider to attack. The target named may be either a target class, defined in the mover's ATK-PRIORITIES entry, or one of the targets explicitly named in the ATK-PRIORITIES table.
- NOW TERRAIN-FOLLOW-AT
  The altitude at which the mover follows terrain may be changed with this entry. A new altitude which differs from that defined in the SDB's BOUNDARY item is specified as:

```
NOW TERRAIN-FOLLOW-AT 300.0 (FT)
```

which would cause terrain following to occur at a height of 300 feet. This option will enable terrain following regardless of the MOVE-OPTIONS setting for this player's tactics. The original SDB value for the terrain following altitude may be reset with:

```
NOW TERRAIN-FOLLOW-AT ORIGINAL ALTITUDE
```

- NOW STOP TERRAIN-FOLLOW
  This entry stops a mover from following terrain, irrespective of the MOVE-OPTIONS setting for this player's tactics.
- NOW EVALUATE-AFTER
  When a plan is being carried out which does not involve attacking a target then it may be necessary to periodically 'wake up' the thinker to consider the current situation. The conditions that are to be considered are those present in the plan and are current at the time the wake up occurs, which may not be the plan in which the wake up was scheduled. This action is useful to check such things as the amount fuel left or distance travelled and so on.
- NOW PRINT NEW-PATH
  This entry causes Suppressor to note the changed flight path in the model output listing, it is useful for monitoring the path of a mover.

Some of the above actions require further tables to define the manoeuvres that will be made in detail. These are the PATTERN, PROFILE and ASPECT action statements which will now be described in more detail.

Plan Patterns

In general, a pattern is a sequence of manoeuvres which may be either repeating or non-repeating. Each pattern is specified by a PLAN-PATTERNS table which has a format consisting of an initial list of all the patterns used by the mover followed by the patterns themselves. Each pattern contains a header with a format like similar to the following:

```
X (M) Y (M) Z (M) REF SPD (M/SEC) TURN-RADIUS (M) DIR
```

although the units specified in the bracketed phrases may be varied. (See the reference manual for details.) The X and Y values refer to the starting point of the manoeuvre relative to those given in the SDB, so that if the initial values of X and Y are both zero then the pattern will begin exactly at this point. The altitude is given by Z and may be referred either to ground level, mean sea level, relative to current altitude, or even relative to a target's altitude, this selection being specified in the REF column.

The points to be traversed in the pattern are tabulated under the heading. At least two points must be specified in order to provide a meaningful path. The path consists of several legs which join the specified points. The legs are travelled at the speed given under the SPD heading, after which an optional column SPD-REF may be given. This specifies how the speed is expressed, the default being in absolute terms ABS. Alternative formats being relative to the mover's current speed, i.e. the speed of the mover at the beginning of the leg, or relative to the target's speed. Upon reaching the next way-point a turn is made on to the next leg with the minum turn radius specified by the TURN-RADIUS column. The final entry indicates the direction of the turn which may be either LEFT or RIGHT. If the pattern is non-repeating the last direction should be labelled as STRAIGHT or STOP, otherwise the pattern will repeat.

An example pattern which is flown by the AEW player-type is:

```
PLAN-PATTERNS
DIMENSION 1 PATTERN-TYPE racetrack
X (KM) Y (KM) Z (M) REF SPD (M/SEC) TURN-RADIUS (M) DIR
   0.0    0.0  9000.0 MSL   175.0            4500.0  RIGHT
   0.0  100.0  9000.0 MSL   175.0            4500.0  RIGHT
  40.0  100.0  9000.0 MSL   175.0            4500.0  RIGHT
  40.0    0.0  9000.0 MSL   175.0            4500.0  RIGHT
END PLAN-PATTERNS
```

consisting of a rectangle with a 100 km long axis pointing north-south and a 40 km long axis aligned east-west. The pattern is flown in a clockwise direction at a constant height of 9000 m above sea level and speed of 175 ms^{-1}; turns with a radius of at least 4.5 km are made at the corners. The pattern would repeat indefinitely without explicit intervention from some other component of the player's MOVE-PLANS.

Plan Profiles

An important part of the 'move_to_attack' plan is the PLAN-PROFILE 'air_intercept', given below, which determines how the interceptor manoeuvres during its attack. Examining a portion of this profile will reveal the structure of such items.

```
PLAN-PROFILE
 DIMENSION 1 PROFILE-NAME air_intercept
  DIMENSION 2  2D-DIST-REL-TGT (KM) 0.0  5.0 10.0 30.0 100.0
   DIMENSION 3  ALT-REL-TGT (KM)  -5.0 -1.0  0.5  5.0
     DIST (KM) ALTITUDE (M) SPD (M/SEC) REF TURN-RADIUS (M)
        12.0   -1000.0     275.0       REL/TGT 1900.0
         0.0       5.0     275.0       REL/TGT 1900.0
        -9.0     500.0     265.0       REL/TGT 1900.0
     DIST (KM) ALTITUDE (M) SPD (M/SEC) REF TURN-RADIUS (M)
        15.0     400.0     275.0       REL/TGT 1900.0
         0.0       0.0     275.0       REL/TGT 1900.0
        -9.0     250.0     270.0       REL/TGT 1900.0
     DIST (KM) ALTITUDE (M) SPD (M/SEC) REF TURN-RADIUS (M)
        12.0     500.0     275.0       REL/TGT 1900.0
         0.0       5.0     275.0       REL/TGT 1900.0
        -9.0     200.0     285.0       REL/TGT 1900.0
   DIMENSION 3  ALT-REL-TGT (KM)  -5.0 -1.0  0.5  5.0
     <Omitted specification for range  5 to  10 km>
   DIMENSION 3  ALT-REL-TGT (KM)  -5.0 -1.0  0.5  5.0
     <Omitted specification for range 10 to  30 km>
   DIMENSION 3  ALT-REL-TGT (KM)  -5.0 -1.0  0.5  5.0
     <Omitted specification for range 30 to 100 km>
END PLAN-PROFILE
```

The format is tabular and is quite similar to that employed by PLAN-PATTERNS, but specifies an extra set of DIMENSION statements due to the fact that the profile is varied as the range from the target changes. The complete entry for the profile is given above only for the range band of 0 to 5 km. Examining the entries we notice that the DIST (distance) entry may be negative, which describes the path of the attacking mover after the approach is complete and the player is receding from the target. The first DIMENSION statement lists the name of all the profiles in the current PLAN-PROFILE block, in this case just the one, 'air_intercept'. The second DIMENSION phrase, one for each profile, lists a set of possible initial distances from the target when the profile is commenced. Here these values range from 0 to 100 km, these values always being positive. Further DIMENSION phrases follow for each such interval describing the range of initial altitude differences between the player and its target. In our example the altitude range is from -5 to 5 km and is divided into three bands. Finally, for each possible combination of initial range and altitude difference a flight profile is specified. For example, if initially the target was 8 km away and 1200 m higher than the mover the first of the profiles listed will be selected. In this case the interceptor will move to a point 10 km away from the target and 1000 m lower than it; it will then close to pass exactly 5 m above the target continuing to a point 500 m above and 9 km beyond the target.

Plan Aspect

The PLAN-PROFILE statement does not give complete control of the movement of an attacking player in Suppressor. Only the height of the player at each range is specified, and not the angular orientation of the two combatants. This will be determined automatically by Suppressor using its own internal logic, (such as the path computed by the setting of the intercept mode.) If this is not satisfactory then a complete specification of the path of the attacking player may be achieved with the PLAN-ASPECT command, which is used in conjunction with the NOW-USE ASPECT option of the MOVE-PLANS block. So for example, we may have the line:

```
NOW-USE ASPECT  air_intercept_aspect
```

with the associated PLAN-ASPECT defined as:

```
PLAN-ASPECT
 DIMENSION 1 ASPECT-TYPE air_intercept_aspect
  ANGLE (DEG) DIR  DIST (KM)  Z (M) Z-REF SPD (M/SEC) SPD-REF
                                             TURN-RADIUS (KM)
    45.0      RIGHT   12.0   -1000.0 REL/TGT  275.0   REL/TGT    1.9
    30.0      RIGHT    0.0       5.0 REL/TGT  275.0   REL/TGT    1.9
   135.0      LEFT     9.0     500.0 REL/TGT  275.0   REL/TGT    1.9
END PLAN-ASPECT
```

This aspect defines the same sequence of heights and ranges as the first entry of the PLAN-PROFILE 'air_intercept'. However, the angular orientation of the two players is also given, and so the distance is never negative; instead, the fact that the aspect angle is greater than 180° indicates that the final entry is a point at which the attacker has passed the target. This table is more powerful than a PLAN-PROFILE but note that several PLAN-ASPECT tables for differing initial heights and ranges cannot be defined. Instead the conditional clauses of the MOVE-PLANS block would need to be used if this flexibility were still desired.

Reactive Movement using Move Plans

We can now synthesize the above discussions to construct a complete MOVE-PLANS procedure for a mover. Each set of MOVE-PLANS consists of one or more named plans listed within separate PLAN blocks. The names of these plans must have been specified under the MANEUVERS entry of the UAN. Each plan consists of a sequence of commands which are either conditional clauses, namely WHEN, BUT-WHEN or OTHERWISE, which may be used to identify the player's current state; or the commands may be action statements, which determine how the player is to act at that moment. Each plan may call other plans, or itself, or may terminate. If the plan is being followed along a pre-programmed path and is completed the next plan within the SDB PATH statement will be followed. If all plans have been completed the player is removed from the scenario and is treated as if it were destroyed by Suppressor.

A plan may use and pass arguments; an argument may be used anywhere a user specified name given in the UAN is used. The arguments themselves must be listed as MANEUVERS in the UAN. An argument may be given an initial value within a plan or on the plan's first mention in the SDB. Suppressor provides no mechanism to distinguish between real arguments, which are the names of the real entities passed between the plans on execution, and the dummy arguments (or parameters), which are their names used within the plans' in their specifications. As such we recommend first that the names of the dummy arguments be identified with the aid of a comment within the UAN. For example, instead of listing manoeuvres as follows:

```
MANEUVERS
  begin_path      enroute      dive_tracker
  going_down      dive         where_next
  back_to_top     reloiter     hang_around
  all_guys        threat_list  next_plan
  charlie         loitering    buzz_around
  go_north/south  sam_guys     at_destination
  back_to_top     priority
```

where all the various types of manoeuvres and their dummy arguments are randomly intermingled, it would be better to define:

```
MANEUVERS
$ PLANS
  begin_path      enroute        going_down
  hang_around     at_destination buzz_around
$ PROFILES
  dive_tracker    dive           back_to_top
$ PATTERNS
  go_north/south  loitering      reloiter
$ CHECKPOINTS
  charlie
$ TARGETS
  all_guys        sam_guys

$ ARGUMENTS
$ Plans
  Where_next     Next_plan
$ Targets
  Priority       Threat_list
```

For clarity the argument names have been spelt using capital letters. Furthermore, the nature of each of the manoeuvres and the arguments have been clearly identified. This will make deciphering a plan a much more straightforward task.

Each plan name has only local scope within the MOVE-PLANS block; this means that the plan names may be reused for different player-types without fear of confusion. So two different player-types may have plans named 'going_down' which perform quite different actions. As a corollary it should be clear that a plan may only call another plan defined within the same MOVE-PLANS block, i.e. a plan belonging to the current player-type. Furthermore, as the physical position of the checkpoints are determined within the SDB independently for each mover many different 'checkpoint charlies' can exist for a single player-type, and even more within the whole scenario. It is the capacity to initialize each mover within the SDB which makes the arguments and plans of the MOVE-PLANS block so useful. Within the SDB each player's mover may be given an initial speed and heading, a plan name (so two movers need not start by following the same plan) and possibly different initial arguments. Their subsequent paths may well be quite different, even though they are using the same block of plans. This discussion will be continued in more depth in the section describing the SDB, however some mover's thinker may include an instruction of the form:

```
PLAN enroute ( all_guys )
```

whereas another may employ:

```
PLAN enroute ( sam_guys )
```

so being directed against a different set of targets albeit with the same plan. These differences can then be used to influence the tactics employed in the RESOURCE-ALLOCATION blocks, which may test to see if the selected target is a member of the mover's active attack priority class and if the mover is currently moving to engage the target. (This being accomplished using the criteria ACTIVE-ATTACK-PRIORITY and MOVING-TO-ENGAGE.)

In the two simple plans described previously for an AEW player no arguments were used. When present they must be contained within parentheses which are themselves completely surrounded by spaces or a terminal carriage return. This form must be followed both in the TDB and the SDB, so that in either data-set the entry:

```
PLAN enroute ( sam_guys)
```

has illegal syntax. The Suppressor parser will be unable to differentiate the closing parenthesis from the word adjoining it.

As an example of the use of arguments let us look at the first two plans, 'begin_path' and 'enroute', for some example player-type:

```
PLAN begin_path END-PLAN
PLAN enroute ( Priority )
  WHEN SNR-STATUS IS DETECT
    THEN:
      WHEN TGT-TYPE IS   sam_battery
        NOW-USE PROFILE  dive_tracker
        AND
          EXECUTE PLAN going_down ( enroute Priority )
      OTHERWISE
        NOW-USE PROFILE  dive
        AND
          EXECUTE PLAN going_down ( enroute Priority )
    END-THEN.
  OTHERWISE
    FOCUS-ON PRIORITY Priority
END-PLAN
```

the first plan, 'begin_path', is a simple place-holder for a SDB pre-programmed path. Its function will be to follow a sequence of predefined points in the SDB until the start point of the plan 'enroute' is reached. Here the sequence of commands contained in this plan will be followed with the argument 'Priority' being replaced by a real target class, either 'all_guys' or 'sam_guys' depending on the relevant SDB entry.

In the above plan, pairs of action statements are linked with the AND statement. In conditional statements this is a logical operator, but here it simply specifies that more than one action is to be carried out. Sometimes the actions are carried out simultaneously, sometimes sequentially with which is which being determined by context. Consider, for example, the plan 'post_launch' below. This is designed to bring a fighter aircraft on-station from its home base:

```
PLAN post_launch
  NOW-USE PROFILE      fly_to_orbit
  AND GOTO POINT       charlie
  AND NOW-USE PATTERN  intercept_orbit
  AND EXECUTE PLAN     await_orders
END-PLAN
```

Here four action statements are unconditionally executed. Initially, the PROFILE 'fly_to_orbit' is used to fly from the airbase to a checkpoint 'charlie'. (Here charlie will be treated by the profile as a 'target'.) Once the checkpoint is reached, the fighter goes into a holding pattern 'intercept_orbit' whilst simultaneously executing the plan 'await_orders'.

The plan 'await_orders' is the second plan in the fighter's tactical manoeuvres, and is:

```
PLAN await_orders
  WHEN SNR-STATUS IS DETECT
    NOW-USE PROFILE air_intercept
    AND EXECUTE PLAN move_to_attack
  BUT-WHEN TOTAL-TIME > 55.0 MIN
    GOTO POINT home
    AND EXECUTE PLAN going_home
  OTHERWISE
    FOCUS-ON PRIORITY air_targets
END-PLAN
```

Summarising the above plan, it first instructs the fighter player-type to fly in a repeating pattern, 'air-intercept', for no more than 55 minutes when it must return home using the plan, 'going_home', should it have detected no targets. If, before this time is up, its thinker assigns a target for engagement, (the SNR-STATUS IS DETECT condition is true in this case) either enemy bombers or fighter-bombers, it should move to attack them using the plan 'move_to_attack'. (This plan will allow the interceptor to continue its mission beyond fifty-five minutes if it is in an engagement as this time elapses, if this behaviour is not desired the order of the initial clauses would have to be reversed.) It knows which targets to look for since it is told with the FOCUS-ON PRIORITY instruction to search for targets which are members of the 'air_targets' class defined previously with the instruction:

```
ATK-PRIORITIES
  DIMENSION 1 LIST-NAME air_targets
    TGT-ELEMENTS bomber fighter/bomber
END ATK-PRIORITIES
```

Turning to the final plan for moving to attack:

```
PLAN move_to_attack
  WHEN SNR-STATUS IS DETECT
    NOW-USE PROFILE     air_intercept
    AND EXECUTE PLAN    move_to_attack
  BUT-WHEN SNR-STATUS IS LOSE-DETECT
    GOTO POINT          charlie
    AND NOW-USE PATTERN intercept_orbit
    AND EXECUTE PLAN    await_orders
  OTHERWISE
    FOCUS-ON PRIORITY   air_targets
END-PLAN
```

we see it is quite similar to the 'await_orders' plan. The main difference is that instead of the alternative WHEN clause directing the fighter home after fifty-five minutes a clause which tests for loss of detection of the target is substituted. Target loss will occur if the target was destroyed, or if it evaded the attack. In this case the instructions are to resume the original 'await_orders' plan. The OTHERWISE statement is a required entry but here just reminds the fighter of the targets it is seeking.

3.6.4 The Tactics of Co-ordination

The final tactical class to be considered is that of co-ordination, the tactics which control how players work together and relay orders and intelligence from one to another. All players are arranged in one or more command chains which define the order of battle of each side in the scenario. These are defined with the help of the COMMAND CHAIN command in the SDB, and as such may be varied from one scenario to another without changing the TDB. This flexibility is slightly spoiled by the requirement Suppressor gives to identify the command chain of player-types which can receive lethal assignments in the TACTIC block of the TDB. This command is the ASG-CMD-CHAIN command, which, for example, has the format:

```
ASG-CMD-CHAIN  sam_chain
```

This indicates that the current player-type is a template of a member of the 'sam_chain' command chain. The role of this item is to ensure that correct decentralization of command occurs when a commander is destroyed. In this eventuality the subordinate will first look for an alternative commander using the ALT-CMDRS: entry in the SDB. If no such commander is available then the player will assume command by changing its own lethal mode of control. However, it cannot do so unless its ASG-CMD-CHAIN entry matches that of the dead commander. Thus, the ASG-CMD-CHAIN entry is necessary for any player who can assume command during a scenario. In the example described this would include both the player seeking out a new commander and any listed commanders in the ALT-CMDRS: entry of the SDB.

A second tactic associated with co-ordination is that describing the ZONE-CHARACTERISTICS. These characteristics describe the ability of a player in the specified zones to report intelligence and send and receive messages to its commanders.

```
$ AEW ZONE-CHARACTERISTICS
$ Co-ordination of players
ZONE-CHARACTERISTICS
  DIMENSION 1 ZONE-TYPE aew_command_zone warn_zone
    ZONE-PERMISSION[20]  SNR-RPT-OK    MSG-RPT-OK
    ZONE-PERMISSION[21]  MSG-RPT-OK    SNR-RPT-OK[22]
END ZONE-CHARACTERISTICS
```

[20] This permission is associated with the first zone in the list.
[21] This permission is associated with the second zone in the list.
[22] The ordering of the permissions is unimportant.

In the example above, the items are a list of zones and only a single DIMENSION statement is required followed by the items whose values are to be set. Here these are the two zones 'aew_command_zone' and 'warn_zone'. Each of these zones will cover some geographical extent, (which is defined in the SDB). Should this table either be absent or the player-type be outside the zone then the information will not be reported. In this case tactics should be present to enable the player to act independently. A differentiation between two forms of data may be made, viz. material gathered by the player's own sensors and information obtained from received messages. If the ZONE-PERMISSION is set for SNR-RPT-OK then data sensed with the player's own sensors will be reported. If communicated intelligence, or the report of the destruction of a player, is to be relayed then MSG-RPT-OK also needs to be set.

Further blocks identify to whom the intelligence is to be reported. The first, MSG-RPT-GUIDE, deals with the reporting of incoming messages and of targets it has seen die:

```
MSG-RPT-GUIDE
  DIMENSION 1 CMD-CHAIN-TYPES  intell
    REPORT-RESPONSIBILITY      CMDR
END MSG-RPT-GUIDE
```

Again a sequence of items form a list and only a single DIMENSION statement is required. This time the listed items are the appropriate command chains that the information may be reported along. In this case only one command chain is present, the 'intell' chain. (The players which compose command chains are again defined in the SDB.) The REPORT-RESPONSIBILITY indicates to whom in the chain the information may be passed, three values are possible:

- CMDR, report up to the player's commander,
- SUB, report down to the player's subordinates, or
- BOTH, report both up to the commander and down to the subordinates.

The next tactical block, SNR-RPT-GUIDE, has a similar function and syntax; its role is to itemize to whom information gathered by the player's own sensors is to be passed. Here the format is slightly more complex, however, due to the fact that it is possible to discriminate between information received from different sensors. This is an example of a table with multiple lists, the first list being the available command chains and the subsequent lists the relevant sensors. In the example there needs be only one second order list, (only one command chain being identified). Furthermore, since only one sensor type is identified only one REPORT-RESPONSIBILITY is required. Again the reporting may take one of three values: CMDR, SUB or BOTH. In the case of the AEW craft the block is quite simple since only one command chain and one sensor receiver are involved:

```
SNR-RPT-GUIDE
  DIMENSION 1 CMD-CHAIN-TYPES  intell
    DIMENSION 2 SNR-TYPE       aew_radar_rx
      REPORT-RESPONSIBILITY    CMDR
END SNR-RPT-GUIDE
```

A further tactical block, INTELL-REPORT-FREQ, is used to specify the maximum rate at which intelligence reporting may occur. For the AEW player-type it is it set as:

```
INTELL-REPORT-FREQ
  DIMENSION 1 CMD-CHAIN-TYPES  intell
    RPT-RATE (1/SEC)           0.0333
END INTELL-REPORT-FREQ
```

which indicates that reporting along the 'intell' command chain cannot occur more frequently than at thirty second intervals.

More complex tactical plans for co-ordination may well be used in many situations. The 'thinker' on board the AEW craft may well have the authority to respond to detected threats by assigning other players to respond to them. For example, if the AEW craft detects incoming fighter-bomber aircraft the commander may order fighters to intercept them. In analysing the incoming intelligence, both from the AEW craft's own sensors and from other sources, various factors may be used in determining appropriate responses. The algorithm used by Suppressor is that when new data arrive their confidence value is compared to that of the stored data, and the new data will replace the old should this confidence value be at least as large as that of the stored data. Normally Suppressor gives all data the same degree of confidence, regardless of their source or age so new data will always oust the old perceptions. This behaviour may be modified with the instruction `INTELL-CONF-FACTOR`, allowing differing degrees of confidence to be attributed to data arriving from different sources; furthermore retained data may be aged so that their confidence value decays with time. This is done with a simple exponential decay so that:

$$\boxed{C(t) = C(0)\, e^{-\alpha t}}$$

where, $C(t)$ is the confidence retained after the initial confidence value, $C(0)$ has decayed at a rate of α for time t measured in seconds.

Three kinds of perceived data may be modified with the above instruction:
- target track, i.e. target position and velocity,
- target `IFF` (Identify Friend or Foe), i.e. hostile, friendly, neutral or unknown, and
- target type, i.e. what sort of player is the target.

The format of the table is represented by the following example:

```
$ This table merely sets the default values of the
$ the confidence in data from the AEWC player
INTELL-CONF-FACTOR
  DIMENSION 1 PLAYER-TYPE  aewc_player
    DIMENSION 2 SNR-TYPE   aewc_radar_rx
      DIMENSION 3 INFO-TYPE TRACK IFF TARGET-TYPE
        CONF-FACTOR         DECAY-FACTOR
          9.0                 0.0
        CONF-FACTOR         DECAY-FACTOR
          9.0                 0.0
        CONF-FACTOR         DECAY-FACTOR
          9.0                 0.0
END INTELL-CONF-FACTOR
```

which does nothing more than confirm the default values for all three data types for the AEW craft's radar receiver. Note that the maximum value of the confidence factor is 9.0, and negative decay rates are not allowed, (so that the stored data may not improve with age).

The table SENSOR-CONF-FACTOR differs from an INTELL-CONF-FACTOR table only in not needing to specify with which player the perception originates. It is therefore a special case of the INTELL-CONF-FACTOR dealing with perceptions which originate with the thinker's own sensing systems, as can be seen below:

```
$ This table sets the default values of the confidence
$ values for the AEWC player's sensor receiver
SENSOR-CONF-FACTOR
  DIMENSION 1 SNR-TYPE   aewc_radar_rx
    DIMENSION 2 INFO-TYPE TRACK IFF TARGET-TYPE
      CONF-FACTOR         DECAY-FACTOR
        9.0                   0.0
      CONF-FACTOR         DECAY-FACTOR
        9.0                   0.0
      CONF-FACTOR         DECAY-FACTOR
        9.0                   0.0
END SENSOR-CONF-FACTOR
```

In the example considered earlier, of an airborne commander ordering fighters to take-off to intercept incoming aircraft, we have used a special command chain that is required to launch subordinate players into movement. In this case an enabling tactical instruction LAUNCH-CMD-CHAIN is required for all players in this command chain. Its form is:

```
LAUNCH-CMD-CHAIN
  aew_command_chain
END LAUNCH-CMD-CHAIN
```

where the listed command chains, in this example their is just one, are those that the current player-type appears in that can be used to launch subordinates into motion.

Maxima may be set for the number of attempts that will be made to send a message with MAX-MSG-ATTEMPTS, and the number of perceived locations that a player may handle at once with MAX-SNR-PERCEPTIONS. If no guide is given to Suppressor in either of the cases then only one attempt will be made to send any message and each player can remember an unlimited number of perceived locations. When limits are set they will be of the form:

```
MAX-MSG-ATTEMPTS         4  (NO-UNITS)
MAX-SNR-PERCEPTIONS     12  (TGTS)
```

When a subordinate's messages to its commander can no longer get through the subordinate may eventually take over command responsibility itself. This will occur after a time interval defined with the `COMM-LOSS-DECENT-TIME`, for example:

```
COMM-LOSS-DECENT-TIME  10.0 (MIN)
```

This would allow the subordinate to take over local responsibility if ten minutes elapse since the first failed communication attempt. If this entry is absent the subordinate will never assume responsibility.

3.7 TDB Summary

The TDB is the largest, most important and most complex of the data sets used by Suppressor in describing a scenario. As a consequence a large portion of this guide has been devoted to outlining the instructions that make up the TDB. Following the completion of the TDB a Suppressor study will progress to either incorporating the environmental information which describes the terrain in which the scenario takes place or directly to the scenario information itself. If terrain information is required the next section describes how it may be incorporated into the simulation. If terrain information is not needed section 5, which outlines the Scenario Data Base or SDB can be proceeded to directly.

4. Terrain Related Instructions

Suppressor has the ability to model the physical environment within which a scenario takes place. In particular the topography of the scenario's surface influences the routes flown by aircraft and the visibility of communication and sensing devices. So when terrain information is included effects such as of the shadowing of radar sensors by hills and valleys and the ability of aircraft to dip low to avoid being sensed can be included. Shadowing of sensors will be modelled automatically within sensing chance computations, and an effective radius for the Earth can be defined for each sensor. Terrain avoidance and terrain following by movers must be explicitly programmed into the scenario's design, but when present will be executed automatically.

In the absence of terrain information Suppressor simply assumes that the Earth's surface is smooth, in other words a uniform sea surface. In these cases a horizon, corresponding to the distance of true horizon for an observer at a given player's height, can still be introduced.

Terrain data are required in the Digital Terrain Elevation Data (DTED) level 1 format, as defined by the US Defense Mapping Agency (DMA). These data[1] consist of the elevations of a points forming a grid covering (in each file) one degree square of the Earth's surface. A special Suppressor processing step, (the DMA instruction set), is provided to convert these data into an intermediate form which may then be used by the Environmental Data Base (EDB) of Suppressor. The output of this stage is then in a form which can be understood by the SDB in constructing a Suppressor scenario.

In the following sections both of the DMA and EDB instruction sets will be discussed, along with an example of how terrain data may be included in a Suppressor model.

[23] DTED format is described in detail by the US military specification MIL-D-89020.

4.1 The Defense Mapping Agency Data Base

The DMA stage processes one or more auxiliary files of Defense Mapping Agency terrain data into a binary terrain file. Generally, this instruction set requires by far the largest amount of disk storage space in a Suppressor scenario, with each data file being typically three megabytes in size. A single DTED file describes one square degree of the Earth's terrain as an array of heights stored at a proscribed resolution. The data are always separated by 3″ of latitude, which corresponds to a ground distance of about 0.05 nautical miles or just under 100 m. The resolution of the data along each parallel of latitude varies with the latitude, thus maintaining an approximately uniform ground spacing of 100 m across most of the Earth's surface. The resolutions are defined in nine bands covering the complete globe as shown in the following table:

Lower Latitude	Upper Latitude	Longitudinal Spacing	Ground Spacing
0°	50°	3″	90 m
50°	70°	6″	120 m
70°	75°	9″	90 m
75°	80°	12″	90 m
80°	90°	18″	100 m

The column headed with 'Ground Spacing' gives the approximate maximum distance along the ground corresponding to these grid resolutions. This will always occur in that part of each band which lies nearest to the equator. For example a spacing of 6″ corresponds to about 120 m along the 50th parallel of latitude but is reduced to about 60 m at the 70th parallel.

To describe the complete environment of a scenario many one degree square terrain files will be needed. For example suppose that terrain describing the whole of Australia is required, spanning from 112°30′ East to 155° East and from 45° South to 10°15′ South. This region would require some 1155 files each containing a square degree of level 1 DTED. The first step in collating these files (assuming the computational resources are available) is to create 35 files each describing one degree of parallel. This is accomplished with the DMA step which builds terrain files which each describe a strip of terrain spanning one degree of latitude.

The entire DMA data file for collating the data for a single parallel of the above example is:

```
EXECUTE
INSTRUCTIONS-FOR:    DMA
$  Collate the data along the 11th parallel South
  LATITUDE-BAND       11
  LONGITUDE-RANGE    112 TO 155
  LONGITUDE-INCREMENT 3 SECONDS
  FILE-CODE-RANGE     31 TO  63

END DMA
END-INSTRUCTIONS DO-NOT-SAVE-DATA
```

Unlike the numerous instructions in the TDB and SDB instruction sets, the DMA only requires five entries of which the first is a comment. The next two items, `LATITUDE-BAND` and `LONGITUDE-RANGE` define the parallel to be described and the longitude range to be covered respectively. Since the `LATITUDE-BAND` entry defines the southern limit of the parallel the example entry

```
LATITUDE-BAND       11
```

specifies the region from 11° South to 10° South. Note that the whole of this band must be included even though the scenario only requires information to 10°15′ South. In short individual DTED files cannot be subdivided but rather the whole file must be included in the database. A similar situation holds at the region's western boundary, since the longitude range is set as

```
LONGITUDE-RANGE    112 TO 155
```

which includes the whole of the region east of 112° East, not just that which lies east of the originally required meridian of 112°30′ East. The two commands specify that 33 DTED files which describe the one degree wide parallel of latitude with a southern boundary of 11° South running from 112° East to 155° East will be incorporated into a resulting terrain strip.

The final two instructions of the DMA file specify the resolution of the data along the arc and the (FORTRAN) logical file units used to read the data. The resolution must be the finest resolution used by any strip in the entire zone to be mapped, not just that appropriate to the current strip. So for example the entry would still be

```
LONGITUDE-INCREMENT 3 SECONDS
```

even for a strip south of the 50th South parallel if it were required to be merged with data north of this parallel. As for the unit numbers since there are 33 DTED files incorporated in the terrain strip and by convention the first file's unit number is 31 the last unit number required will be 63 so giving the range:

```
FILE-CODE-RANGE      31 TO  63
```

Similar data base files are required for each of the strips in the whole zone, 35 in all this case. Once the DMA step has been carried out for each of the 35 strips these strips can then themselves be collated with the Environmental Data Base (EDB) step to produce a single output terrain file describing the entire region.

4.2 Environmental Data Base

The next step in running a simulation involves processing the EDB instructions, which specify the entire area of a scenario and its resolution limits. The binary files created by processing the DMA data-base serve as the input for the EDB processing. During this step terrain data are decomposed into equilateral triangles covering the entire area of the scenario. Initially, the EDB processor uses large triangles to cover the entire surface of the terrain. A deviation threshold value is then used to determine if the resulting triangulation is sufficiently accurate. If it is not the larger triangles may be decomposed into smaller triangles until the resulting triangulation describes the terrain with sufficient accuracy. This can result in large triangles being retained over some parts of the scenario while in other areas the terrain would be specified by smaller triangles.

Like the DMA, the EDB consists of a limited number of instructions. In addition to a comment line, it requires instructions giving the latitudes and longitudes of the most extreme points in the input terrain files and the specifications for the largest and smallest sizes of the triangles in units of the distance between DMA data points. These values are given as logarithms to the base two. This means that a maximum triangle value of 6 implies that the largest triangles will have sides with lengths 64 times greater than the distance between the original DTED points. (And thus span approximately 6 km on the ground.)

The EDB instruction set for the example of a large zone covering the whole of Australia mentioned earlier is:

```
EXECUTE
INSTRUCTIONS-FOR:
EDB
An EDB file covering the whole of Australia
TERRAIN MASKING
  MIN/MAX LATITUDE:    4500000S  1000000S
  MIN/MAX LONGITUDE: 11200000E 15500000E
  MIN/MAX TRIANGLE:     4    6
  DEVIATION THRESH:    50.0 M
END EDB
END-INSTRUCTIONS DO-NOT-SAVE-DATA
```

The MIN/MAX LATITUDE gives the southern and the northern boundaries of the terrain strips processed in the DMA step. The format used requires that that the minutes, seconds and tenths of seconds must be included in the value. These will be always zero in the EDB, since the underlying DTED files always lie on whole degree boundaries. Similarly the MIN/MAX LONGITUDE entry gives the western and the eastern edges of the terrain strips. The final two data items specify how Suppressor will triangulate the DTED data to produce terrain data which can be used within the model. As noted above the MIN/MAX TRIANGLE values refer to the sizes of the smallest and largest triangles to be formed and these are in powers of 2. Clearly the larger these values the coarser the resulting triangulation and the smaller the resulting terrain file. For a region as large as the example shown above the triangulation should be quite coarse to prevent the final terrain file being impracticably large. The final entry, DEVIATION-THRESHOLD defines the threshold value Suppressor uses in determining whether or not to decompose larger triangles into smaller ones. It is expressed in metres as a floating point number and describes the maximum allowed difference between any DTED point's altitude and the triangulation's representation of the terrain's altitude at that point. The smaller this threshold the smaller the resulting triangles and the more disk space required by the terrain data.

The file output by the EDB stage is known as an "untranslated binary terrain file". It can be read in (and translated) by the SDB step of the Suppressor simulation run. As long as the region covered by different scenarios does not change many different scenarios can use the same data file. The intermediate DMA data files and the original DTED data files would not have to be retained if no further terrain processing is required, so freeing up large amounts of disk space.

4.3 Generating Terrain Data

Because the DMA and EDB steps are relatively straightforward it has been possible to provide users of Suppressor at the AOD an automatic method to generate terrain files. It was noted in Section 2 of this guide that terrain related files are to be found under each model's directory structure in the 'dma', 'edb' and 'sqr' directories respectively. As may be expected the DMA and EDB instruction files are in the 'dma' and 'edb' directories respectively, along with the output DMA terrain strip files and the output EDB untranslated terrain file. The 'sqr' directory holds the initial DTED files, each covering one square degree. To generate the required EDB data file first ensure that all the required DTED files are present in the 'sqr' directory. These files must be given unique names which identify the south-western corner of each file. These names are formed as follows: the first two characters of the file name are the digits giving the grid square's latitude, with the third character being either 'n' or 's' to indicate the northern or southern hemisphere respectively. The next three characters give the longitude of the cell's south-western corner, and the final character of the file's prefix is either 'w' for the western hemisphere or 'e' for the eastern hemisphere. (All cell's lie entirely within one hemisphere or another, so the determination of north or south and east or west is never ambiguous.) Finally the file name must be completed with a '.dt1' suffix. Some examples are:

File Name	Latitude & Hemisphere	Longitude & Hemisphere
23n148e.dt1	23° North	148° East
02s017w.dt1	2° South	17° West
00n179e.dt1	0° North	179° East
53n180w.dt1	53° North	180° West
89n000e.dt1	89° North	0° East

Note that since the corner chosen is the south-western corner of the cell the file 00s000w.dt1 would be doubly impossible, and instead the file should be called 00n00e.dt1 since the cell's data all lie in the northern and eastern hemispheres of the globe.

Once this is done a single subsequent file 'square' must be created in the 'sqr' directory. This file contains four lines in any order specifying the extent of the zone, the minimum and maximum triangle sizes and the deviation threshold. For the example described above covering Australia we would have:

```
latitude    -45  -10
longitude   112  155
triangle      4    6
deviation    50.0
```

The keywords `latitude`, `longitude`, `triangle` and `deviation` must be present. All values except the deviation threshold must be integers, the deviation threshold may be a real. The units (which must be metres) of the deviation threshold do not need to be specified but they may be. Once this information is provided the 'run' shell script will generate all terrain information automatically when it is required by a scenario.

4.4 Terrain Summary

The two optional DMA and EDB processing stages produce an untranslated binary file which contains the information that Suppressor needs to incorporate terrain into a scenario. This file can be interpreted by the SDB processing step to produce a translated terrain file for the final MOD processing stage of Suppressor. In order to use the AOD 'run' facility correctly in the presence of terrain information the SDB command option which is used to describe terrain information, which is one of `DO-NOT-USE-TERRAIN`, `USE-TRANSLATED-TERRAIN` or `TRANSLATE-TERRAIN` option must be given as a comment in the MOD file, i.e the SDB entry

```
TRANSLATE-TERRAIN
```

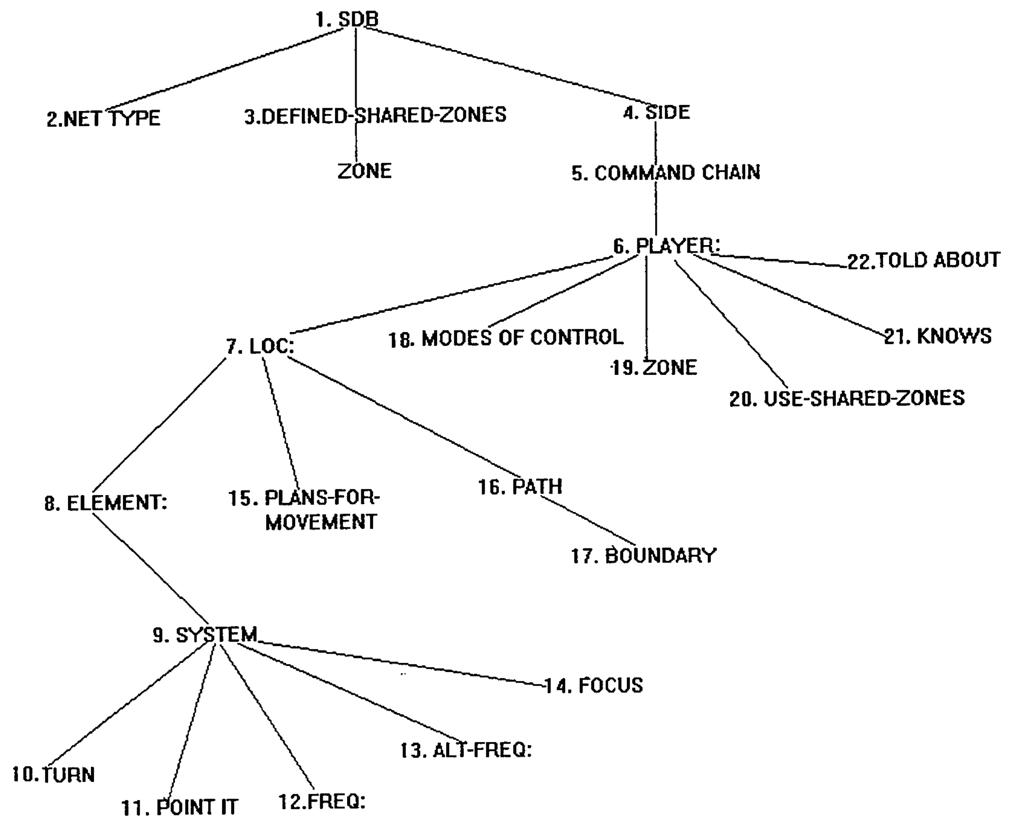
would be also given in the MOD file as:

```
$ TRANSLATE-TERRAIN
```

This is not a requirement of Suppressor but merely of its local implementation.

5. Scenario Data Base

The Scenario Data Base (SDB) is used to define all the data which are specific to a particular Suppressor scenario. Many different scenarios may use the same TDB instruction set, since the player-types used and the tactics they employ do not vary. The SDB allows the deployment of the forces to be varied, as well as their order of battle. In addition small local changes to portions of the TDB may be embedded in an SDB file when required. (In this way minor variations of a player-type's behaviour which are specific to a particular scenario can be kept with that scenario's definitions.)



The overall structure of the SDB data set is illustrated in the above diagram, which shows the hierarchical structure of the data set. In the first portion of the file, immediately following the SDB keyword, are the keywords defining the global properties of the scenario, such as its size and position. Then three further data structures are used to define communication nets, the scenario's geographical zones, and both the order of battle and the deployment of the forces taking part in the scenario.

The SDB commands which are used to define each side in the scenario contain the information describing each command chain; the number and deployment of each player-type; and the initial and future locations of all the elements employed in the scenario. In addition to the initial positions of moving players all their pre-programmed movements and the operational characteristics of all players' component systems are defined. This information includes such things as the frequencies of communication receivers and the types of communication networks they operate on, as well as the operational availability of weapons, sensors and disruptor systems.

5.1 Global SDB Instructions

The SDB instruction set, as with the other major instruction sets, commences with the keywords EXECUTE and INSTRUCTIONS-FOR: on separate lines followed by the entry SDB. These items are then followed by a mandatory comment. Just as with the TDB data set this comment may be more than one line long, but it must always be present. Again, as with the TDB data set, these comments do not need to be preceded by the normal comment identifier, '$'. The comment is terminated by the keyword RANDOM-NUMBER-SEED:, which is the first mandatory instruction of the SDB data set. Six mandatory commands accompanied by two optional commands always commence the SDB instruction set. These must always be given in the order shown below:

```
SDB
This comment line is mandatory
RANDOM-NUMBER-SEED: 98345215
RADIUS OF SCENARIO:      1500.0 (KM)
LOCATION RESOLUTION TIME:   1.0 (SEC)
START TIME:                 0.0 (SEC)
STOP TIME:                  2.4 (HR)
CHECKPOINT TIME INCREMENT: 6.0 (HR)
CENTER OF SCENARIO L/L:  3749400S  14454400E
USE-TRANSLATED-TERRAIN
<omitted SDB instructions>
END SCENARIO COMPLETE
```

The RANDOM-NUMBER-SEED: item is used to initialize a random number generator, the seed is a positive odd integer, and would normally have eight or nine digits. The entire scenario should take place inside a circle with a radius specified by the RADIUS OF

SCENARIO: entry. This means that not only should no players be located or even be able to move outside this circle but also that they not be allowed to interact with entities lying beyond this radius. The circle whose radius is defined by this item is centred on the origin of Suppressor's Cartesian coordinate system, i.e. its position is given by the ordered pair (0.0, 0.0) in X and Y coordinates. This coordinate system can be referred to a point on the Earth's surface by using the later, optional, CENTER OF SCENARIO L/L: item. This places Suppressor's origin of coordinates at the specified latitude and longitude, which is given as a pair of strings of unpunctuated digits followed by one each of either 'N' or 'S' and 'E' or 'W' standing for north, south , east and west respectively. The digits represent first degrees, then minutes, then seconds and finally tenths of seconds. In the example above the centre of the zone is 37°49'40.0" South latitude and 144°54'40.0" East longitude, roughly the location of the AOD at Fisherman's Bend in Melbourne. When the CENTER OF SCENARIO L/L: item is absent the scenario is assumed to lie on the equator at the prime meridian, which will be of no consequence if terrain information is not being used.

The accuracy with which Suppressor models the location of moving players is determined by the LOCATION RESOLUTION TIME: entry. If set to zero then Suppressor will use the maximum possible accuracy and recompute locations of all elements involved in interactions whenever these interactions occur. If this value is larger than zero Suppressor will re-use computed positions which are closer in time than the value defined by this entry. So in the above example which uses a time of 1 s any player's computed position could be used for an interval of two seconds, one second either side of the moment for which the position is exact. Note that Suppressor has only been tested with this value less than 5 s. The START TIME: command specifies the earliest possible time at which any event may happen within the scenario; it should never be negative. Normally zero will be chosen but the value may be positive provided no players are defined to begin moving before this moment. The STOP TIME: is when the model ceases. Since Suppressor models chains of events the model will effectively cease once all the events have terminated, so this time may be any which is larger than the time of the last event in the scenario. The CHECKPOINT TIME INCREMENT: is used to set the interval at which Suppressor will write out a backup file containing enough information to allow a restart of the scenario at this point. (Note that here checkpoint is being used with a different meaning than when used with manoeuvres.) If this time is zero or greater than the STOP TIME: the only checkpoint file is that produced at the end of the run; a smaller value will produce intermediate files which can be used to monitor the progress of a long running model.

The first non-mandatory item CENTER OF SCENARIO L/L: is used to locate the scenario on the Earth's surface and was discussed above. When this option is specified the type of terrain information that is present must be noted. At this stage terrain data may still be absent, and if so this is specified with the keyword DO-NOT-USE-TERRAIN; this instruction can also be used to force Suppressor to disregard existing terrain data. If terrain data is to be used then either data that has been already translated by a previous SDB processing pass is used, with the USE-TRANSLATED-

TERRAIN option; or untranslated terrain direct from the terrain processing stages is used, when the TRANSLATE-TERRAIN option must be given.

If navigational errors are to be modelled using the NAV-ERROR-DATA option of the TDB then the MAG-DIP-ANGLE: item must be used:

```
MAG-DIP-ANGLE:      6.8 (DEG)
```

to define the local inclination of the Earth's magnetic field. This entry is not otherwise required.

5.2 Communication Nets

Whenever any players within the scenario are meant to be able to communicate with each other then the communication nets which they will use must be defined using the NET TYPE item of the SDB. Broadly speaking, the NET TYPE data item specifies the types of communication networks that might be employed in a scenario, the types of messages to be transmitted over each network and the time delays and priorities associated with each type of message in each network.

Communication corresponds to the talking generic function, as noted in the TDB section of this guide, and may employ either implicit or explicit communication nets. Implicit nets use landlines and these cannot be jammed and they do not require that the players communicating across them be in the line-of-sight of each other. Explicit nets use broadcast RF (radio frequency) signals; these may either be jammed or their signals can be blocked by intervening terrain. (So an explicit net cannot be used by two players stationed on opposite sides of a mountain range.) Atmospheric degradation factors, defined by the TRANSMISSION LOSS TDB data item may also be important to the operation of explicit nets. Communication systems using implicit nets do not actually need to have any physical capabilities defined, (being in effect simple telephones), but any device defined as a RF communication device for use on explicit nets may also be used on implicit nets. The extra physical capabilities of the explicit communication systems being simply disregarded.

Each communication net that is to be used requires a corresponding NET TYPE entry, so that the explicit communication net 'uplink' declared in the UAN as

```
EXPLICIT-NETS  uplink
```

could for example be defined as follows:

```
NET TYPE  uplink  MODE:  INTERMITTENT
                  CHANGE FREQ DELAY:  0.5 (SEC)
  MSG WPN-ASGN    TRANSMIT-TIME:    0.19 (SEC)
                  1-WAY PRIORITY: 82
  MSG MOC-CHANGE  TRANSMIT-TIME:    0.21 (SEC)
                  1-WAY PRIORITY: 82
  MSG CNCL-ASGN   TRANSMIT-TIME:    0.26 (SEC)
                  1-WAY PRIORITY: 76
  MSG ASGN-STAT   TRANSMIT-TIME:    0.32 (SEC)
                  1-WAY PRIORITY: 68
  MSG SUB-CUING   TRANSMIT-TIME     0.43 (SEC)
                  1-WAY PRIORITY  60
  MSG ENG-STAT    TRANSMIT-TIME:    0.21 (SEC)
                  1-WAY PRIORITY: 54
  MSG DEATH       TRANSMIT-TIME:    0.24 (SEC)
                  1-WAY PRIORITY: 35
  MSG MOVE-ORDER  TRANSMIT-TIME:    0.14 (SEC)
                  1-WAY PRIORITY: 20
  MSG INTELL      TRANSMIT-TIME:    0.18 (SEC)
                  1-WAY PRIORITY: 11
```

The first part of this entry identifies the net in question with the name 'uplink', which is specified as an explicit net in the UAN. (Note that no part of the `NET TYPE` entry discriminates between implicit and explicit nets, this being done in the UAN entry instead.) The keyword `MODE:` is used to indicate how messages are transmitted across the net. When, as with uplink, the net is `INTERMITTENT` mode, then transmitters on the net are operating only when a message is being passed. In the alternative `CONTINUOUS` mode transmitters on this net will be permanently broadcasting a signal. A net in `CONTINUOUS` mode is much more susceptible to detection by a warning receiver than one in `INTERMITTENT` mode.

If a communication receiver in use on the net switches to an alternative frequency in order to avoid jamming then this operation will take the amount of time specified by the `CHANGE FREQ DELAY:` entry. If a receiver has no alternative frequencies available to it this entry will be disregarded.

After the basic characteristics of the net have been described the `NET TYPE` entry lists the kinds of messages that may be sent across this set and sets a transmission time and a priority for each of them. For example the entry:

```
MSG MOVE-ORDER  TRANSMIT-TIME:    0.14 (SEC)
                1-WAY PRIORITY: 20
```

specifies that that a message relaying movement orders will take 0.14 s to transmit across the net and have a relative priority of 20. If all the messages processed by a net have equal priority then messages will be routed in order of their reception. By setting different priorities messages likely to be of great importance, like a weapon assignment order, can be pushed up the queue. The higher the numerical value of the priority the higher the priority of the message. So, for example, all messages with a priority of 40 will be dealt with before messages with priorities of 39 will be sent.

In all there are nine possible types of messages, which are as follows:

- ASGN-STAT[24], assignment status;
- CNCL-ASGN, cancel assignment; to subordinate
- ENG-STAT, engagement status;
- DEATH, report of death of player;
- INTELL, intelligence;
- MOC-CHANGE, change the lethal mode of control of a subordinate;
- MOVE-ORDER, movement order;
- SUB-CUING, cuing order to subordinate, (which sets the reference 'heading' of a non-moving subordinate which thereafter acts as the reference angle from which all angles around this subordinate will be measured);
- WPN-ASGN, weapon assignment message; used to send assignments to subordinates.

Care should be taken in defining a communication net to ensure that the players using the net can send the types of messages they need to exchange in order to function properly. In short the tactics which involve message passing are those of lethal assignment, lethal mode of control, launch movement orders and intelligence reporting. The first of these three categories correspond to the four LETHAL-ASSIGNMENT procedures, the GUNS-FREE/GUNS-TIGHT procedures and the LAUNCH-START procedure in RESOURCE-ALLOCATION tactics. The fourth category is not linked to explicit procedural tactics, but rather to the presence of a player on an intelligence reporting command chain. The different message types are associated with the different tactical abilities of a player as follows:

- LETHAL-ASSIGNMENT requires: WPN-ASGN to make assignments to a subordinate; CNCL-ASGN to cancel the assignments; ASGN-STAT to report the status of an assignment or ENG-STAT to report the status of any consequential lethal engagement. Finally if the RESOURCE-ALLOCATION procedures employ the SDB-CUING-FOR-LOC or TGT-CUING-FOR-LOC options then the net must pass SUB-CUING messages.
- GUNS-FREE/GUNS-TIGHT requires MOC-CHANGE to change the lethal mode of control of a subordinate.
- LAUNCH-START requires MOVE-ORDER to issue an order to start a subordinates motion.
- Intelligence reporting requires INTELL and DEATH to pass information across a net.

Each communication net must have at least one such message description, since only those messages so described may be handled by that net. On occasions when many

[24] ASGN-STATUS and ENG-STATUS are in fact synonyms and may be used interchangeably.

messages are sent the available nets will tend to become swamped and messages may backlog substantially.

5.3 Scenario Zones

Once communication nets have been defined the next optional entry is the keyword DEFINE-SHARED-ZONES which is used to establish zones which one or more players will use. The zones are the same as those referred to in each player-type's ZONE-CHARACTERISTICS item in the TDB. They are used to restrict the attention of the player to certain geographical regions whilst making tactical decisions. For example, a RESOURCE-ALLOCATION block may only allow targets to be selected for a particular player when the targets lie within certain zones. The actual location of these zones is determined within the SDB either by a ZONES entry in the PLAYER: item of the SDB or, when several players need access to the same zones, under a DEFINE-SHARED-ZONES entry. The syntax used within the ZONES and the DEFINE-SHARED-ZONES entries is identical, and an example is:

```
DEFINE-SHARED-ZONES
 ZONE 1 sam_zone     STATIONARY
  MIN/MAX ALT: 0.0   (M)  MSL   1500.0    (M)  MSL
    X,Y:  440.0 -180.0 (KM)  X,Y:  440.0 -140.0 (KM)
    X,Y:  480.0  -60.0 (KM)  X,Y:  480.0   60.0 (KM)
    X,Y:  380.0  160.0 (KM)  X,Y:  380.0  220.0 (KM)
    X,Y:  220.0  320.0 (KM)  X,Y:  100.0  320.0 (KM)
    X,Y:   20.0  220.0 (KM)  X,Y:   20.0   80.0 (KM)
    X,Y:  160.0  -20.0 (KM)  X,Y:  160.0 -100.0 (KM)
    X,Y:  240.0 -180.0 (KM)
END DEFINE-SHARED-ZONES
```

Here a single shared zone, 'sam_zone' is declared which will be shared by several players within the SDB. This zone is STATIONARY which means its co-ordinates, listed below as ordered pairs, are in absolute units referred to the scenario's origin. These co-ordinates must be listed in a sequence which traverses the zone's boundary, and either an anti-clockwise or a clockwise ordering may be used. Zones can also be declared as RELATIVE, which allows them to defined relative to some location, which may be a moving location and is almost certain to be different for different players, a useful feature when shared zones are being used. For example consider:

```
DEFINE-SHARED-ZONES
 ZONE 1 offset_triangular_zone     RELATIVE
  MIN/MAX ALT: 0.0   (M)  REF   100.0   (M)  REF
  RELATIVE TO PLAYER  10 aew_player LOC: 1
    X,Y:  440.0 -180.0 (KM)  X,Y:  300.0  -80.0 (KM)
    X,Y:  240.0 -180.0 (KM)
END DEFINE-SHARED-ZONES
```

which defines a triangular zone defined relative to `LOCATION` 1 of player 10 of the SDB, which is of the 'aew_player' type of the TDB. Notice that with this form of a zone definition a player will know that it is inside the zone, (when tested in a `RESOURCE-ALLOCATION` procedure for example), only if it knows where the zone's centre, i.e. the reference player is. Hence the player must be either a known and perceived target or a perceived friendly player at the time of the determination. When a zone's boundaries are defined in terms of a reference position the altitude limits of the zone, specified by the `MIN/MAX ALT` entry, may be qualified using the `REF` keyword, which implies that these altitudes are in terms of the reference location's height. Alternatively the altitude can be defined in terms of mean sea level, `MSL`, or as height above ground level, `AGL`.

In addition to defining a zone in terms of a player's location it is also possible to refer a zone to a `CHECKPOINT` position known to the player or relative to an explicitly defined point. So for example we may have either:

```
RELATIVE TO CHECKPOINT  charlie
```

or

```
RELATIVE TO X,Y,Z:  37.5 103.0  (KM)   150.0 (M) AGL
```

The point may also be specified in terms of latitude and longitudes, with these being given using the same format as the centre of the scenario:

```
RELATIVE  TO  L/L,Z:   04754120N   13412450W    150.0  (M)
MSL
```

specifying a point 150 m above sea level at 47°54'12" N and 134°12'45" W for example. Note that the finest resolution (of one tenth of an arc second) would imply a ground resolution of about 3 m, more than adequate for most needs and far more accurate than the resolution of any level one DTED terrain information.

The final entry of the `ZONE` item may be either a list of points which define the zone's perimeter, (as in the above examples), or a description of a circular zone. In the latter case the range of radii of the zone plus the angular range must be specified, for example:

```
MIN/MAX RNG:  10.0  100.0   (KM)
```

specifies an annular zone ranging from 10 km to 100 km from the zone's centre. The angular extend of a circular zone may also be defined, for example the limits

```
COUNTERCLOCKWISE FROM:  0.0 (DEG) TO:  -90.0 (DEG)
```

defines a zone ranging from due east (the reference direction) to due south and ranging through 270° of arc. If the keyword `CLOCKWISE FROM` had been used the zone would have been the complement of this one, and so range through 90°.The default reference direction of due east may be changed by qualifying a `RELATIVE TO CHECKPOINT` or `RELATIVE TO X,Y,Z:` instruction with the `HDG:` statement:

```
RELATIVE TO CHECKPOINT  charlie
HDG:  45.0 (DEG)
```

The above makes the reference location point north-east for example. If the zone is defined with respect to a player's location the heading of the selected element at this location will determine the zone' reference direction.

5.4 Deployment of Forces

The numbers and deployment of forces amongst the sides which feature in any scenario is determined by the next SDB data block, the `SIDE` entry. At least one such item must appear in any scenario. The role of the `SIDE` instruction is to locate all the players taking part in the scenario, determine their command and control relationships with one another and describe the movement plans that these players have. Although any scenario must have at least one `SIDE` defined within it, typically two will be present but many sided conflicts may also be modelled. In this way conflicts involving three or more sides can be studied using Suppressor. Each `SIDE` defined in the SDB must have been declared in the `SIDES` entry of the UAN. So a combatant side 'Orange' may be defined within a `SIDE` entry with the heading:

```
SIDE Orange
  <Omitted commands>
END SIDE
```

whilst a neutral side may also be included in a scenario with the aid of the keyword `NEUTRAL`, so for example:

```
SIDE Swiss NEUTRAL
```

Each side within a Suppressor scenario is made up of one or more chains of command, each explicitly modelled with a `COMMAND CHAIN` data item. This defines all the players which are members of the command chain and the chain of command amongst these players. Each player is introduced with the `PLAYER:` command, which, if the same player has already been defined in the SDB, is just a tag showing the player's name and identification number along with its position in the command chain. (A single player may be in several command chains and be at a different level of command in each of them.) At the top of each chain is a commander with a command level of 1, for example:

```
COMMAND CHAIN  sam_chain
  PLAYER:   1 command_base  LEVEL:  1  END PLAYER
END COMMAND CHAIN
```

A command chain may have more than one such head, for example

```
COMMAND CHAIN  sam_chain
  PLAYER:   1 command_base  LEVEL:  1  END PLAYER
  PLAYER:   2 western_sam_hq  LEVEL:  1  END PLAYER
END COMMAND CHAIN
```

would introduce two autonomous top level commanders. Below the commander may be several subordinates players at the second level of command:

```
COMMAND CHAIN  sam_chain
 PLAYER: 1 command_base  LEVEL:  1  END PLAYER
   PLAYER: 11 early_warning_radar  LEVEL: 2 END PLAYER
   PLAYER: 12 sam_player  LEVEL:  2  END PLAYER
   PLAYER: 13 sam_player  LEVEL:  2  END PLAYER
 PLAYER: 2 western_sam_hq  LEVEL:  1  END PLAYER
   PLAYER  21 sam_player  LEVEL  2 END PLAYER
END COMMAND CHAIN
```

The command base has players representing two SAM sites, numbers 12 and 13, under its command along with a long range radar site. The next player in the list, '2 western_sam_hq', is at the same command level as the command base so it is not under the command of the base, and further players in the list will lie under the command of this player. The command chain shows that the second commander has a single SAM player, '21 sam_player', under its command. Each SAM site may have further players under their command, for example:

```
PLAYER   12 sam_player   LEVEL:  2  END PLAYER
  PLAYER  121 sam_missile
    LEVEL:  3  (FOR DISAGGREGATION ONLY)
  END PLAYER
```

would place a SAM missile 121 under the command of the SAM player 12. This missile is a `FUTURE-PLAYER` which will be 'disaggregated' when launched, as indicated by the `(FOR DISAGGREGATION ONLY)` option of the `PLAYER:` item. A future player may not have further real players under its command but it may have further future players which may in turn disaggregate from it. For example the command chain fragment:

```
PLAYER      10 air_base  LEVEL:  2  END PLAYER
  PLAYER  101 f18_fighter  LEVEL:  3
    (FOR DISAGGREGATION ONLY)
  END PLAYER
    PLAYER 1011 ARM_missile  LEVEL:  4
      (FOR DISAGGREGATION ONLY)
    END PLAYER
```

shows an air-base commanding an F18 aircraft which it launches as a future player, this player in turn may launch a n ARM (anti-radiation) missile as a future player.

With disaggregated future players only one such player need be specified in the `COMMAND CHAIN` entry but several future players may in fact be created, the actual number being limited by the `PLAYER-STRUCTURE` definition of its player-type which disaggregates the future players as a resource. The first future player launched will be labelled with the identification number in the SDB, e.g. the first ARM missile fired will have the label 1011. Missiles launched subsequently will have these labels incremented by unity, so that the next ARM missile will have label 1012 and so on.

It should be noted that there is no need for the player's labels to be unique across the whole scenario as long as they are unique to the players with that particular type. For example no two players of the 'f18_fighter' type should have the same label.

5.5 Player Initialization

The function of the PLAYER: item in the examples so far has been just to identify each player's position within the order of battle. However, each player must be defined in detail on its first occurrence in a COMMAND CHAIN. The detailed definition is used to locate the player, determine its pre-programmed motion (if any) and initialize the player's state. This initialization is made with the PLAYER: item, an example being:

```
PLAYER: 12  sam_player   LEVEL: 2
$ LOCATION 1 contains the headquarters and a radar system
  LOC:   1  X,Y,Z: 140.0    90.0 (KM)   0.0   (M)   AGL
    HDG: 45.0 (DEG/N/CW)
    ELEMENT:  10   sam_hq  DISCRETE QUANTITY: 1
      SYSTEM:  1020  comm_rx ON
        FREQ:  2.0 (GHZ)  NET: 63 broadcast
      SYSTEM:  1022  radio_rx ON
        FREQ:  4.3 (GHZ)  NET: 64 uplink
    END ELEMENT
    ELEMENT:  11   sam_radar  DISCRETE QUANTITY: 1
      SYSTEM: 1130  sam_radar_rx ON
        FREQ:  1.5 (GHZ)
    END ELEMENT
  END LOCATION
$ LOCATION 2 contains a SAM battery
  LOC:  2  X,Y,Z:  100.0  -20.0 (KM)  0.0 (M)  AGL
  END LOCATION
$ LOCATION 3 contains a SAM battery
  LOC: 3   X,Y,Z:   80.0   40.0 (KM)  0.0 (M)  AGL
  END LOCATION
  MODES-OF-CONTROL:  ENGAGE  command_base
  USE-SHARED-ZONE 1 sam_zone
END PLAYER
```

The SAM site player '12 sam_player' is here defined in full. It consists of four elements distributed over three physical locations although only two of these elements need explicit initialization. Once the player's role in the current command chain is defined with the LEVEL entry each location is initialized in turn. The first location is set up with the commands following the LOC: 1 entry. The first function of this item is to define the location's position, done here in the scenario's absolute co-ordinate system. (Latitude and longitude could also be specified.) None of the SAM sites are mobile so all the locations are fixed, but a 'heading' may be specified for fixed sites with the HDG: entry. Here the item

```
HDG: 45.0 (DEG/N/CW)
```

sets the heading to be north-east, (45° clockwise from north). Mathematical orientations of anti-clockwise from due east may also be set with the (DEG) and (RAD) angular measures. Changing the heading in this way will effect the reference vectors for angular measures in such systems as sensors and relative zones. This entry will have little effect on moving locations, i.e. those locations that contain a mover, since here the heading is always the 'natural' heading of the mover. (Movers which start moving after an explicit start-movement order will commence their motion with this heading if one is provided, otherwise they will initially head due east.)

5.5.1 Initialization of Elements

Attributes of all elements found at each location may be varied with the `ELEMENT:` item. For example the headquarters element is initialized above as:

```
ELEMENT:  10   sam_hq  DISCRETE QUANTITY: 1 CRITICAL
  SYSTEM:  1020  comm_rx ON
    FREQ:  2.0 (GHZ) NET: 63 broadcast
  SYSTEM:  1022  radio_rx ON
    FREQ:  4.3 (GHZ) NET: 64 uplink
END ELEMENT
```

Here the survivability of the element is set with the keywords `DISCRETE QUANTITY: 1 CRITICAL`, which, in this case, override the settings of `DISCRETE 1 NONCRITICAL` that had been set for this element in the TDB.

The `SYSTEM:` item may be used to set whether a system is initially `ON`, `OFF`, or `NON-OP`, which is to say on, off or non-operational. The default is `ON` except for tracking sensor receivers which are initially `OFF`. The `TURN` command may also be used to turn systems on or off or make them non-operational at any moment during the scenario, so that

```
TURN ON AT TIME:  1.0 (HR)
TURN OFF AT TIME: 1.5 (HR)
```

would turn on a system for thirty minutes. (Do not, however, use this item to set a system's initial status.) Note that this item has only intended for use with disruptors, communication devices and sensors. Its effectiveness with movers, thinkers and weapons is not guaranteed.

Another important qualifier within the `SYSTEM:` data item is the `POINT IT` keyword. This is used to orient a system in a specific direction and is useful for communication devices on explicit nets, sensors and explicit disruptors. If not specified these systems will point in the direction of their location's heading or, in the case of sensors, rotate. Two variants of the `POINT IT` instruction exist for use with sensor receivers; the first specifies some particular direction in which to point the sensor:

```
POINT IT IN ABS DIR AZ,EL: 135.0  23.4  (DEG) TARGET
```

the example would point the receiver in an azimuthal direction of 135°, (normally north-west), and at an elevation of 23.4° at almost all times, even if the system is located on a moving vehicle with a changing heading. The one exception will be if the sensor receiver is used as a tracking receiver in order to track a target. The system's orientation can be completely fixed, so that it will not even track a target, by using the keyword `FIXED` instead of `TARGET`. A direction relative to a variable heading may also be specified with the keyword `REL`, as follows:

```
POINT IT IN REL DIR AZ,EL:  -34.0  0.0  (DEG)  FIXED
```

Alternatively a system may be pointed at a specific location, so that if the item

```
POINT  IT  AT  LOCATION  X,Y,Z:   143.5   12.3  (KM)   165.0
(M)
```

would force the system to always point at the specified point. (Which may be given as a latitude and longitude.) The ability to point systems is very useful for modelling RF transmission and reception accurately, since it is the only way to force Suppressor to compute the strengths of incoming and outgoing signals with all portions of an antenna pattern. (Otherwise all these systems are assumed to be using their mainbeams most of the time.)

The operating frequencies of sensors and communication devices are set with the `FREQ:` entry of the `SYSTEM:` data item. For radar systems the transmission frequency of the system (set by the `XMIT-FREQ` or the sensor transmitter) may be over-ridden by using this option either with the sensor transmitter or the sensor receiver, (but not both unless they agree). In our example above we have:

```
SYSTEM: 1130  sam_radar_rx ON  FREQ:  1.5 (GHZ)
```

which ensures that this radar is operating at 1.5 GHz, regardless of the setting in the TDB[25]. For optical sensors this entry is used with the sensor receiver to set the frequency used by the receiver, but its only function is in determining the correct pass-band for any `TRANSMISSION-LOSS` table that may be present for the system.

[25] It is probably not a good idea to use this feature too much, since the performance of a receiver, i.e. its antenna gain and beamwidth, are not in fact independent of the operating frequency of the system.

The FREQ: keyword is essential to determine the operating frequency of communication systems operating on explicit nets. It is coupled with a NET: keyword which indicates which communication net the communication receiver is operating on. One device may be operated on several nets. The above example indicated that the receivers '1020 comm_rx' and '1022 radio_rx' operated at frequencies of 2.0 GHz and 4.3 GHz on nets '63 broadcast' and '64 uplink' respectively. These nets must have been defined with a NET TYPE command in the SDB, (as the example above for nets of type 'uplink'). A different example where one receiver is operated on more than one net is:

```
SYSTEM:  1020  comm_rx ON
  FREQ:  2.0 (GHZ) NET: 63 broadcast
  FREQ:  4.6 (GHZ) NET: 65 uplink
```

Here the explicit net '65 uplink' is not the same communication net as '64 uplink', even though it has the same type. (It will, however, have the same characteristics as '64 uplink'.)

When creating a Suppressor scenario it is very important that all the frequencies used by the different communication devices are in agreement, otherwise the systems will not function correctly. Suppressor provides no global variables so these values must be entered separately but correctly for each system using a communication net. Furthermore note that the operating frequency of communication transmitters is not set; they are linked to specific receivers and are assumed to operate on the same frequencies as these systems.

If a sensor system is able to respond to perceived jamming by changing its frequency, (as set for example with the J/N-NOISE-OPERATOR-THRESHOLD) then these alternative frequencies are listed in the SDB as the ALT-FREQ: item. A sensor transmitter with four possible frequencies would be defined as:

```
SYSTEM 1022 radar_rx
  FREQ: 1.0 (GHZ)
  ALT-FREQ: 1  1.3 (GHZ)
  ALT-FREQ: 2  0.8 (GHZ)
  ALT-FREQ: 3  2.1 (GHZ)
```

The three alternative frequencies are identified by an integer label; the initial frequency is effectively frequency number zero. The frequencies will be used in the order indicated and after all have been exhausted the system will cycle back to its initial operating frequency.

For communication systems the ability to evade jamming is conferred by the setting of the J/N-COMM-OPERATOR-THRESHOLD in the TDB and the communication receiver having listed alternative frequencies in the SDB, for example:

```
SYSTEM: 1024 comm_rx ON
  FREQ: 1.4 (GHZ)  NET: 14  broadcast
    ALT-FREQ: 1  1.8 (GHZ)
    ALT-FREQ: 2  1.6 (GHZ)
```

sets a receiver to have two frequencies which are alternatives to its principal frequency. Only one communication system on each net should have alternative frequencies, otherwise there is a danger that different systems will change frequencies at different times and so break off communication links.

Disruptors may be initialized with focused pre-emptive spots already operating when the system is turned on. This is accomplished with the FOCUS data item as follows:

```
SYSTEM:  2050 jammer
  FOCUS PULSE SPOT LO/HI-FREQ:  4.3  4.5 (MHZ)
  FOCUS NOISE SPOT LO/HI FREQ:  5.4  5.5 (MHZ)
```

would focus a pulsed spot with frequencies ranging from 4.3 MHz to 4.5 MHz and a noise spot from 5.4 MHz to 5.5 MHz. The system may emit both noise and pulsed spots regardless of the setting of its DISRUPTOR-CHARACTERISTIC entry PULSE or NOISE, but when a pulsed spot is to be emitted the SUBCARRIER-BANDWIDTH must be defined in the TDB DISRUPTOR-FREQ-LIMITS item.

5.5.2 Player Movement

Once the definitions related to each element are completed the PLANS-FOR-MOVEMENT data item can be given. Its role is to specify the movement plans for players which have no pre-programmed paths but will nonetheless move upon receipt of a start movement order. For example an interceptor which is take off from an airbase may well use the plan 'post_launch' from the MOVE-PLANS discussed in the TDB guide, which made use of a CHECKPOINT 'charlie' which must be defined for each mover which uses this plan, giving a PLANS-FOR-MOVEMENT item of the following form:

```
PLANS-FOR-MOVEMENT
  PLAN post_launch
  CHECKPOINT charlie
    X,Y,Z:  -40.0  260.0 (KM) 3750.0 (M) MSL
      SPD:    265.0 (M/SEC) TURN-RADIUS: 2250.0 (M)
  CHECKPOINT home
    X,Y,Z: -200.0  190.0 (KM) 0.0 (M) MSL
      SPD:    265.0 (M/SEC) TURN-RADIUS: 2250.0 (M)
END PLANS-FOR-MOVEMENT
```

Here the CHECKPOINT is defined along with the player's initial speed and its minimum turn radius. Notice that if the player is disaggregated, i.e. is a FUTURE-PLAYER, then if the speed and turn radius are not specified then default values of 1 ms^{-1} and 1 m will be used. Several points can in fact be associated with each CHECKPOINT, for example we could have defined:

```
CHECKPOINT charlie
  X,Y,Z:  -40.0  260.0 (KM) 3750.0 (M) MSL
    SPD:    265.0 (M/SEC) TURN-RADIUS: 2250.0 (M)
  X,Y,Z:   60.0  280.0 (KM) 3500.0 (M) MSL
  X,Y,Z:  -10.0  230.0 (KM) 3500.0 (M) MSL
    SPD:    255.0 (M/SEC)
```

This implies that the player will move from its initial location to the three specified points in turn, and will slow to 255 ms^{-1} in flying past the last point; note that if no new SPD: entry had been provided the mover would have maintained its initial speed. This sequence will be broken should the player break away on a reactive manoeuvre at any time. The points in these scenarios are all defined in terms of the scenario's absolute co-ordinate reference frame, but as always latitudes and longitudes could have been used.

In the initial PLANS-FOR-MOVEMENT entry a second named CHECKPOINT 'home' is included; this does not occur in the initial 'post_launch' plan but may be found in subsequent MOVE-PLANS plan for this player, e.g. in 'going_home'. Checkpoints which are not in fact used by any of the MOVE-PLANS entries may be defined, provided they are declared within the UAN as MANEUVERS.

Should the initial movement plan expect arguments the actual values of the arguments should be provided in the PLANS-FOR-MOVEMENT; for example if the plan 'enroute' were to be used for the initial routing of the player it would need the actual name of the target class as an argument, for example:

```
PLAN enroute ( sam_guys )
```

which would initialize the named SAM sites in a TDB ATK-PRIORITIES entry as potential targets.

Most players which move will have at least some predefined routes, although they might still move reactively away from these paths when required, (their MOVE-PLANS would need to be able to handle this eventuality). The pre-programmed routes are set up with the PATH instruction, which for an entirely pre-programmed mover may look like:

```
PATH  START TIME: 535.0 (SEC)  ALT: MSL  MODE: 3-D
  PLAN bomb_target ( ground_targets )
    CHECKPOINT charlie
    X,Y,Z:  -480.0   100.0 (KM)   350.0 (M)
      SPD:   295.0 (M/SEC) TURN-RADIUS: 2000.0 (M)
    X,Y,Z:  -420.0   260.0 (KM)   350.0 (M)
    X,Y,Z:  -240.0   250.0 (KM)   800.0 (M)
    X,Y,Z:   -80.0   340.0 (KM)  2400.0 (M)
      SPD:   325.0 (M/SEC)
    X,Y,Z:   100.0   340.0 (KM)  1000.0 (M)
    X,Y,Z:   360.0   160.0 (KM)  1300.0 (M)
    X,Y,Z:   260.0   -80.0 (KM)  1000.0 (M)
    X,Y,Z:  -480.0   100.0 (KM)  6200.0 (M)
```

which represents the motion of a player modelling bomber throughout a scenario. When the PATH item is provided no PLANS-FOR-MOVEMENT entry can be present for the same player, and vice versa. The first line of the PATH entry specifies three mandatory items:

```
PATH  START TIME: 535.0 (SEC)  ALT: MSL  MODE: 3-D
```

these are the time at which the bomber starts to move, (here 535 s into the scenario), the reference for the specified altitudes, given in the example as being above mean sea level, (ground level could also be specified); and finally the mode used by Suppressor to describe the path, here 3-D. The 3-D mode is appropriate for aircraft, since in this mode all changes of direction must be made along smooth arcs which respect the mover's minimum turn radius. Ground vehicles may be modelled with the SURFACE mode, in which all motions are point-to-point and turns occur instantaneously. (Ground vehicles may also be modelled with 3-D mode, but usually their minimum turn radius is small enough that the choice is immaterial, though this may not always be the case with surface ships though.)

Although this player has fully pre-programmed movements it does have a MOVE-PLANS entry, which is actually just to identify the bomber's targets so it can in fact attack them, and it does not alter the movements of the player at all. The TDB entry is:

```
$ This move-plan actually just identifies targets,
$ no movement is included at all
MOVE-PLANS
  PLAN bomb_target ( Priority )
    FOCUS-ON PRIORITY Priority
  END-PLAN
END MOVE-PLANS
```

Once the hook into the player's MOVE-PLANS are made with the PLAN entry a CHECKPOINT 'charlie' is named which corresponds to the first point flown to by the bomber; this CHECKPOINT is an example of a redundant definition since no reference is made to it within the player's MOVE-PLANS, (it could have been omitted if desired). This checkpoint is followed by the player's speed, which must be provided after the first point, and minimum turn-radius, which must also be provided initially if the mode is 3-D. A list of points the bomber will move to in sequence then follows, with the bomber's speed being increased part way through the scenario.

It is possible to attempt to constrain the arrival time of a pre-programmed mover at some point along a path with the TIME-WINDOW option, for example

```
L/L,Z:   2754130S 13212230E  1000.0 (M)
  TIME-WINDOW   45.0 (MIN)   50.0 (MIN)
```

would try and place the mover one km above the point with 27°54'13.0" S latitude and 132°12'23.0" E longitude in between 45 minutes and 50 minutes into the scenario. Suppressor will attempt to adjust the mover's speed in progressing from the previous point to meet this constraint, but the minimum and maximum speed and acceleration limits will be honoured, meaning that the arrival time cannot be guaranteed.

More than one PLAN may be referred to within a PATH command, for example:

```
PATH  START TIME: 7220.0 (SEC)  ALT: MSL  MODE: 3-D
PLAN begin_path
  CHECKPOINT alpha
    X,Y,Z:    -20.0   -440.0 (KM)    850.0 (M)
      SPD: 240.0 (M/SEC)  TURN-RADIUS: 2000.0 (M)
    X,Y,Z:    160.0   -440.0 (KM)    850.0 (M)
    PLAN enroute ( sam_guys )
  CHECKPOINT bravo
    X,Y,Z:    240.0   -460.0 (KM)    850.0 (M)
    X,Y,Z:    320.0   -400.0 (KM)    850.0 (M)
    X,Y,Z:    360.0   -320.0 (KM)    850.0 (M)
    PLAN at_destination
  CHECKPOINT charlie
    X,Y,Z:    500.0   -220.0 (KM)    850.0 (M)
```

Here the mover does have the ability to manoeuvre reactively, but the role of the PATH command here is not just to specify a sequence of way-points for the mover but also to indicate when particular MOVE-PLANS should be executed. Initially, whilst moving to checkpoints alpha and bravo the plan 'begin_path' is in force; subsequent to this the plan 'enroute' is used; when the last way-point in the PATH is reached, CHECKPOINT 'charlie', the plan 'at_destination' takes over.

If any player uses terrain avoidance, terrain following or threat avoidance strategies at any time the BOUNDARY data item is required. This entry is used to define the maximum volume in which the mover can operate. It is completed in two parts, the first follows the PATH statement's first line and specifies the operational height limits of the mover, an example being:

```
PATH  START TIME: 20.0 (MIN)  ALT: MSL  MODE: 3-D
 BOUNDARY VERTICAL LIMIT: MIN 0.5 (KM) AGL MAX 5.0 (KM) MSL
```

where the player is constrained to attempt to remain at least 500 m above ground level and at most 5,000 m above mean sea level when executing its manoeuvres. Normally when following or avoiding terrain the aircraft will remain at the lower altitude of course; recall that this value may be changed in the player's MOVE-PLANS with a NOW TERRAIN-FOLLOW-AT instruction. Note that the minimum altitude *must* be specified as above ground level, AGL, whilst the maximum altitude is *always* referred to mean sea level, MSL.

Each BOUNDARY item must be completed by defining the horizontal bounds of the mover's operational zone, in other words the region within which the mover is constrained to remain. As with the definition of zones this boundary must be specified by traversing the points used to define the bounding region in either a clockwise or anticlockwise order. The boundary can be specified using either Cartesian co-ordinates or the latitudes and longitudes of the relevant points. An example of a complete BOUNDARY specification is:

```
BOUNDARY
  VERTICAL LIMIT: MIN 0.5  (KM) AGL  MAX 5.0  (KM) MSL
  L/L: 3600000S 14000000E  L/L: 3600000S 14500000E
  L/L: 4000000S 14500000E  L/L: 4000000S 14000000W
```

In the above entry the perimeter points are read in one row at a time, not one column at a time.

5.5.3 Modifying Player Tactics

The MODES-OF-CONTROL: data item is used to modify the way a player will evaluate certain RESOURCE-ALLOCATION procedural blocks, in particular those dealing with which player makes the decisions pertaining to launching a player into movement, controlling a lethal or non-lethal engagement, ordering jamming or initiating emission control. In the RESOURCE-ALLOCATION procedures conditions may be tested whose results will depend on who controls these processes, for example

```
IF ENG-CONTROL-MODE IS SELF
```

would be true if the current player has the authority to make decisions about engagements. The MODES-OF-CONTROL: entry is used to determine whether or not the player does indeed have these authorities. So by setting

```
MODES-OF-CONTROL:
  LAUNCH  fighter_player
  ENGAGE  fighter_player
```

the authorities for the current player's actions pertaining to launching and engagement are given to a player of the type 'fighter_player'. (Which may or may not be the current player's type.) This player would never need to consider disrupting or emission control, but if it did the full set of four qualifiers would be needed, as in the example:

```
MODES-OF-CONTROL:
  LAUNCH  fighter_player
  ENGAGE  fighter_player
  DISRUPT fighter_player
  EMCON   CMDR
```

Note that whenever a player uses the RESOURCE-ALLOCATION criteria ENG-CONTROL-MODE, LAUNCH-CONTROL-MODE, JAM-CONTROL-MODE the appropriate MODES-OF-CONTROL: entry must also be included here in the SDB.

Emission control is handled slightly differently than the other MODES-OF-CONTROL; first of all the MODES-OF-CONTROL: option EMCON may be set to be either CMDR, (a commander of the current player), or the name of the current player-type. No criterion to test emission mode of control exists, instead the EMCON/TURN-ON and EMCON/TURN-OFF procedures behave according to whether emission control is held by the commander or if the player is autonomous. When the EMCON mode is CMDR the player will only consider turning on its sensors when an assignment has been received from its commander, and when all assignments have been cancelled the player will turn off its sensors. If the EMCON mode refers to the player itself it will be able to evaluate turning on and off its sensors at all times.

Engage mode-of-control may change during the scenario if one of the following events occurs:
- Death of a commander: a subordinate may assume control;
- Decentralization: a commander may decentralize control to its subordinates;
- Communication jamming, a subordinate may temporarily assume control.
- Receipt of a change of mode-of-control message, that is a CHANGE-MOC, from a commander, (which will be originated by the GUNS-FREE/TIGHT procedures.)

The usefulness of the above feature is that it allows players sharing the same tactics (being of the same player-type) to make different tactical decisions based on their differing circumstances.

5.5.4 Setting Zones

A player may have geographical zones defined for its private use with the ZONE keyword; the syntax is almost exactly like that for DEFINE-SHARED-ZONES which was described earlier. So for example a private 'fighter_patrol_zone' may be defined with:

```
ZONE 1 fighter_patrol_zone     STATIONARY
  MIN/MAX ALT: 0.0   (M)  MSL   1500.0    (M)  MSL
     X,Y:  440.0 -180.0 (KM)  X,Y:  440.0 -140.0 (KM)
     X,Y:  480.0  -60.0 (KM)  X,Y:  480.0   60.0 (KM)
     X,Y:  160.0   60.0 (KM)  X,Y:  160.0 -100.0 (KM)
     X,Y:  240.0 -180.0 (KM)
```

The only difference with the DEFINE-SHARED-ZONES entry is that the zones are simply listed without needing to be enclosed in a distinct block, which is the case with DEFINE-SHARED-ZONES and END DEFINE-SHARED-ZONES.

If instead of using its own private zones a player is to use shared zones then the names of the relevant shared zones must be listed with the USE-SHARED-ZONE command, so that

```
USE-SHARED-ZONE 1 sam_zone
```

would enable a player to use the shared zone '1 sam_zone' defined earlier.

5.5.5 Setting Initial Perceptions

Each player may be given certain perceptions of friendly players (with the KNOWS keyword) and unfriendly players (with the TOLD ABOUT keyword) by the SDB. In both cases these perceptions do not have to be true, which allows the player to hold misperceptions about the scenario. Commanders who will be launching other players into motion must have a KNOWS entry describing these players to them. In addition any intervening players in the command chain between the commander and the player to be moved must also have this information provided to them in a KNOWS entry.

Let us consider the information held by a commander with authority to issue scramble orders to an airbase and to issue orders to friendly fighter aircraft:

```
KNOWS 71 fighter_player O/A
  X,Y,Z: -200.0   190.0 (KM)   0.0 (M)   AGL
KNOWS 341 airbase OP
  X,Y,Z: -200.0   190.0 (KM)   0.0 (M)   AGL
    HAS  1 fighter_player
    HAS  6 fighter_player_alert
```

We see that the player knows that the player '71 fighter_player' is out of action, indicated by the O/A qualifier, and that its current location is as specified, (although remember that this information is not necessarily accurate.) (At this stage of a scenario the out of action property just signifies that the player is not active at the commencement of the scenario but is awaiting orders.) The commander has no further information about this player but it also knows about '341 airbase', in particular that it is operational, OP. Furthermore the commander knows the airbase's location, and, through the information provided by the HAS options, that the airbase has one 'fighter_player' and six 'fighter_player_alert', (FUTURE-PLAYERS which may disaggregate into players of the type 'fighter_players'), available. The information is important when a commander's RESOURCE-ALLOCATION tactics are testing for the presence of a subordinate's resource (for lethal assignment or launching movement for example). Without the KNOWS and HAS keywords the procedure will fail since the commander will not know of the subordinate's available resources.

The final option for the PLAYER: definition is the TOLD ABOUT item, which is the player's initial (briefed) perception of threats. This information may or may not be accurate and is required when threat avoidance is being used by the player. Since a player may occupy several physical locations it will be told about specific LOCATION entries, so for example:

```
TOLD ABOUT 31 sam_player  LOC. 2
  X,Y,Z: 120.0 -100.0 (KM)   0.0 (M) AGL  BY 201 fighter
```

would inform a player about a SAM battery at the specified location; in this case the information came from '201 fighter'. This entry can in fact be the player's own name to indicate that the intelligence originated from the player's own sensors. Note that if a player identified by a TOLD ABOUT option is a potential target for engagement then this player should already have been previously defined in the SDB to avoid potential difficulties for Suppressor in identifying the target correctly.

5.6 SDB Summary

The SDB information is now complete, and in processing this stage the various players and their pre-programmed paths will be computed. This information will be recorded in the SDB's listing file, which should be checked for consistency with expectations. Once this stage is complete the next step in the process, actual model execution, may commence. This is handled by the 'Model Kick-Off Database', the MKO or MOD stage. The instructions and output produced by this stage will be discussed in the next section of the guide.

6. Model Kickoff Instructions

The instruction sets described previously, i.e. the UAN, TDB, EDB and SDB, are concerned with defining the capabilities of the players along with their environment and disposition. The Model Kickoff stage, (the MOD), is where Suppressor brings all these data together and carries out the simulation. As such the main role of the MOD is to produce output files which comprehensively describe the outcome of the scenario which can then be used for future analysis. The MOD produces output in three forms, the first and least important being a short summary of the current state of the simulation which can be produced on the computer's display. More importantly two data files are also produced: the first is a simple chronological listing of all the events which occurred during the scenario. These items constitute the 'Time History Listing File' and is produced as a plain text file which gives a readable, but verbose, history of the scenario. The reference manual contains a complete listing of the various messages that Suppressor can print in this file to indicate the progress of the simulation. Finally a binary file written in an internal Suppressor format may be output. This can be used by the Analysis Data Base, (ADB), to produce a preliminary analysis of the scenario's outcomes and also may be used by other software tools in interpreting the results of the simulation.

The MOD instruction set is primarily designed to offer some control over the form of the output produced by Suppressor, although some limited influence on the scenario itself may still be exercised in the MOD. In this section of the guide the MOD instruction set is discussed along with its role in the simulation. Only a brief overview of the Time History items is given, a more detailed reference to these entries being available in the reference manual.

6.1 Model Execution

The purpose of this procedure is to initialize important variables used in controlling model operations. This is achieved through the use of five initialization entries, four of which are required. These variables and the format for their input is shown in the following example:

```
EXECUTE
INSTRUCTIONS FOR:
MODEL EXECUTION
  RANDOM NUMBER SEED:      654321033
  SIMULATION END TIME:     2.0 (HRS)
  ENVIRONMENT:             OTHER  TERMINAL
  OUTPUT:                  DISPLAY/PRINT/FILE
  SKY RADIANCE:            NIGHT
    <omitted data items>
END MODEL INPUT
```

The MOD instruction set commences, like the other Suppressor data sets, with the two keywords `EXECUTE` and `INSTRUCTIONS-FOR:` on separate lines. These are followed by the entry `MODEL EXECUTION` which identifies the remaining instructions as being part of a model kick off file. Following this item are four mandatory instructions. The first of these is a random number generator seed, for example:

```
RANDOM NUMBER SEED:  11296597
```

The seed should be a positive odd integer consisting of eight or nine digits. It can however be set to zero to ensure the random number seed defined in the SDB input is used.

The next required entry determines when to stop the simulation, for example:

```
SIMULATION END TIME:  360.0 (MIN)
```

Time is usually entered as a positive real number with units as seconds, minutes or hours. However, as with the random number seed, this value can be set to zero allowing the scenario end time specified in the SDB to be retained.

The third required entry, `ENVIRONMENT`, is obsolete and has no effect on either the model or its output but it must still be included. Its original function was to inform Suppressor of the computer environment it was running under. The command is followed by two qualifiers, the first being either `VAX` or `OTHER` and the second either `BATCH` or `TERMINAL`, for example:

```
ENVIRONMENT:  VAX  TERMINAL
```

The last mandatory entry allows the user to specify the form of the output for which there are several options. The output produced during model execution forms the entire record of the simulation and is therefore essential. The output consists of three forms: a real time summary which is displayed on the user's screen during the model run; a chronological list of time history data items which may be written to an ASCII file; and a binary record of the scenario which may be sent to a data file. Which of these output records are produced by Suppressor is controlled with the `OUTPUT` keyword, for example the screen summary will be produced when the `DISPLAY` option is set:

```
OUTPUT:  DISPLAY
```

An ASCII file containing a set of chronological time history data items will be produced if the following instruction is given:

```
OUTPUT:  PRINT
```

The binary data file is produced by the command:

```
OUTPUT:  PRINT
```

All output may be suppressed with the keyword NOTHING. Any combination of output forms may be set by combining the output specifiers as follows:

- DISPLAY/PRINT/FILE, which produces all three forms of output.
- DISPLAY/FILE, which suppresses the Time History listing.
- DISPLAY/PRINT, which suppresses the binary file.
- PRINT/FILE, which suppresses the display screen.

After initializing the required variables there is also one optional entry, SKY RADIANCE. This variable allows the user to specify whether the mission is occurring during the day:

```
SKY RADIANCE:  DAY
```

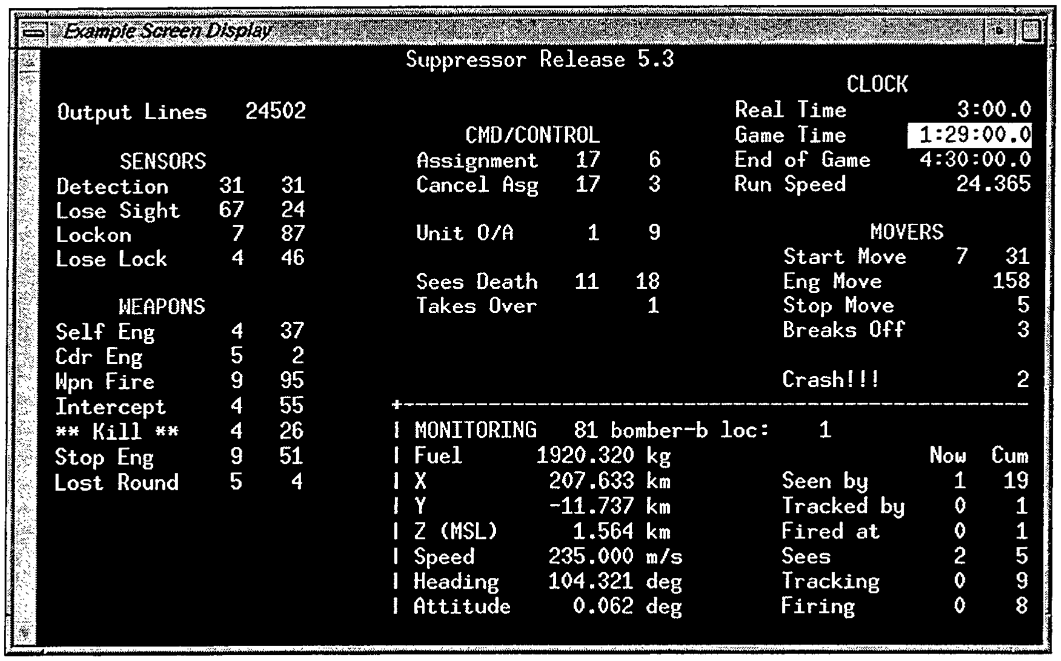
or at night:

```
SKY RADIANCE:  NIGHT
```

By default Suppressor assumes that missions occur during the day. When a night-time mission is modelled the background radiance is reduced so diminishing the contrast of objects when perceived with optical sensors which in turn makes them harder to see. On the other hand the visibility of infrared targets will be increased. (For more details see the discussion of optical and infrared sensor capabilities in section 3.2.2.4 of this guide.)

6.2 Screen Display Output

As already stated there are three forms of output: the display window, a binary file used for input into the ADB and a listing file. Initially consider the screen display. An example window taken during the course of a simulation is given below. The information contained in the window is separated into six sections: four summary sections which are always present are SENSORS, WEAPONS, CMD/CONTROL, and MOVERS; a game CLOCK gives both game time and real elapsed time. The 'Run Speed' item indicates the current rate of progress of the simulation; the value of 24.365 indicates that the 'game', i.e. the simulation, is progressing at a rate where 24.365 units of game time will elapse for every unit of real time. The history of an individual player can be monitored in the MONITORING section in the lower right-hand portion of the screen.



```
                              Suppressor Release 5.3
                                                                 CLOCK
 Output Lines    24502                                   Real Time            3:00.0
                                  CMD/CONTROL            Game Time          1:29:00.0
      SENSORS                Assignment    17     6      End of Game        4:30:00.0
 Detection      31    31     Cancel Asg    17     3      Run Speed            24.365
 Lose Sight     67    24
 Lockon          7    87     Unit O/A       1     9                  MOVERS
 Lose Lock       4    46                                     Start Move      7     31
                             Sees Death    11    18          Eng Move             158
      WEAPONS                Takes Over           1          Stop Move              5
 Self Eng        4    37                                     Breaks Off             3
 Cdr Eng         5     2
 Wpn Fire        9    95                                     Crash!!!               2
 Intercept       4    55    +--------------------------------------------------------
 ** Kill **      4    26    | MONITORING    81 bomber-b loc:     1
 Stop Eng        9    51    | Fuel        1920.320 kg                       Now   Cum
 Lost Round      5     4    | X            207.633 km            Seen by      1    19
                            | Y            -11.737 km            Tracked by   0     1
                            | Z (MSL)        1.564 km            Fired at     0     1
                            | Speed        235.000 m/s           Sees         2     5
                            | Heading      104.321 deg           Tracking     0     9
                            | Attitude       0.062 deg           Firing       0     8
```

In the summary sections a cumulative count of events of a particular type is shown, each column representing a single side taking part in the scenario. Only one column is shown for scenarios with single sides. In scenarios having more than two sides the left hand column will always show the first side listed in the SDB, whilst the right hand column will change as it cycles through the remaining sides one by one. Finally a header line is present which shows the current length of the output listing file, this file being discussed below.

6.3 Data Capture And The Time History Data Listing

The role of the DATA CAPTURE instruction is to select which time history data items are to be written to either the listing file or to the binary file. If no action is taken then Suppressor will list all events occurring in the scenario to each file selected with the OUTPUT entry. This can produce very long data files indeed, making the analysis of the output very difficult since relevant data is buried amongst the many superfluous entries. The use of DATA CAPTURE allows users to focus on only those aspects of the model in which they are interested. DATA CAPTURE takes the form of a list, which must contain at least one entry, of time history selection phrases. The format is as follows:

```
DATA CAPTURE                  PRINT             FILE
<time-history-name>  <first-option>  <second option>
END CAPTURE
```

Each phrase consists of the name of the time history data item (these can occur in any order) followed by two options both of which can take values of either YES or NO. The first option determines whether or not the selected data item is printed in the listing file; the second option controls whether it is recorded in the binary file. An example shows this more clearly:

```
DATA CAPTURE                         PRINT    FILE
  ,NET-BUSY,-WANTS-TO-TALK-TO         NO       YES
  DID-NOT-CHANGE-ITS-PATH             NO       NO
  STARTING-TO-RELOAD                  YES      NO
END CAPTURE
```

Note that since without a DATA CAPTURE entry all events are logged to the data files the default for both these options is YES. Therefore the DATA CAPTURE item need only be used when a selected data item is not required in one or both files.

The time history listing file is the primary source for the evaluation of a completed scenario. An example listing below shows all events occurring during a period a mere 2.2 seconds long some 6 minutes after the start of a simple scenario:

```
6:01.5
10 bomber ADDS-WPN-TO-ENG-QUEUE-FOR-TGT 101 target weapon:
  130 bomber_weapon
6:01.6
11 bomber NOW-SENSING-ELEMENTS-OF-TGT 101 target
radar cross section: 1.0 square meters, at 173.8 degrees
  azimuth;  1.9 degrees elevation; with sensor: 120
  bomber_radar_rx elements: 11 target_ele
6:01.8
11 bomber INITIATES-PERCEPTION-OF 101 target and FIRST-
  DIRECTLY-SEES it
with sensor: 120 bomber_radar_rx; tgt (x,y,z): 19.000 10.500
  0.000 km; at time: 6:01.6; spd: 0.0 m/s; hdg: 90.0 deg;
  sense time: 6:01.6; sensor (x,y,z): -15.3 14.2 1.200 km; 3-
  D dist: 34.6 km
10 bomber NOW-SENSING-ELEMENTS-OF-TGT 202 target
radar cross section: 1.0 square meters, at 172.2 degrees
  azimuth; 1.9 degrees elevation; with sensor: 120
  bomber_radar_rx elements: 11 target_ele
6:02.3
11 bomber NOW-SENSING-ELEMENTS-OF-TGT 202 target
radar cross section: 1.0 square meters, at 172.4 degrees
  azimuth; 1.9 degrees elevation; with sensor: 120
  bomber_radar_rx elements: 11 target_ele
6:03.6
10 bomber NOW-SENSING-ELEMENTS-OF-TGT 201 target
radar cross section: 1.0 square meters, at 171.8 degrees
  azimuth; 1.9 degrees elevation; with sensor: 120
  bomber_radar_rx elements: 11 target_ele
6:03.7
10 bomber NOW-SENSING-ELEMENTS-OF-TGT 103 target
radar cross section: 1.0 square meters, at 173.7 degrees
  azimuth; 1.9 degrees elevation; with sensor: 120
  bomber_radar_rx elements: 11 target_ele
```

As can be seen a very large amount of data is generated even for this very short segment of a simple model, (involving the perceptions of a few targets by two bombers). The amount of output may be substantially reduced and the focus concentrated on the more relevant information with the DATA CAPTURE item, in this case used to suppress the NOW-SENSING-ELEMENTS-OF-TGT. (This item confirms that a sensor has in fact sensed a target.)

```
DATA CAPTURE                              PRINT     FILE
  NOW-SENSING-ELEMENTS-OF-TGT              NO        YES
END CAPTURE
```

With this change to the MOD data set a much clearer listing is obtained for the same period of time:

```
6:01.5
10 bomber ADDS-WPN-TO-ENG-QUEUE-FOR-TGT 101 target
weapon: 130 bomber_weapon
6:01.8
11 bomber INITIATES-PERCEPTION-OF 101 target and FIRST-
   DIRECTLY-SEES it
with sensor: 120 bomber_radar_rx; tgt (x,y,z): 19.000 10.500
   0.000 km; at time: 6:01.6; spd: 0.0 m/s; hdg: 90.0 deg;
   sense time: 6:01.6; sensor (x,y,z): -15.3 14.2 1.200 km; 3-
   D dist: 34.6 km
```

In this listing attention is drawn to the actions and perceptions of the two bombers, i.e. processes mediated by the players' thinkers and their tactical blocks. It is now clearly seen that one bomber has already assigned a weapon to engage the target, whereas the other bomber has only just perceived this target.

There are many possible entries in the Time History listing, each of which has a fixed syntax, which is usually reasonably self explanatory. As noted above a full description of all these entries is to be found in the reference manual. Furthermore the process of obtaining information from the Time History listing may be assisted with the ADB module, which will be discussed in more detail later.

6.4 Debugging of Resource Allocation Procedures

This data item assists in the development of Suppressor models, being designed to assist in the debugging of the `RESOURCE-ALLOCATION` tactical procedures described in section 3.2.4 of this guide. In particular it allows the user to test `RESOURCE-ALLOCATION` criteria to determine the causes of unexpected behaviour. Examination of the output produced by the `DEBUG` command may indicate why the model is not behaving as expected.

There may be more than one `DEBUG` item in each MOD data file but each `DEBUG` command must be identified uniquely with a `RESOURCE-ALLOCATION` procedure from the `TACTIC` block in the TDB, so there may be only one `DEBUG` item for any given procedure. An example `DEBUG` command used to examine the `LETHAL-ENGAGE-FIRING-START` procedure of a player `88 F-16` is:

```
DEBUG LETHAL-ENGAGE-FIRING-START
  FOR-PLAYER  88 F-16
    VS.-TGT  22 F-111
    TIME-WINDOW  15.0  55.0  (MIN)
    USING-WPN  ANYONE
END DEBUG
```

The first two lines of the command identify the relevant RESOURCE-ALLOCATION procedure and uniquely identifies the player about which we are concerned. If more than one player of this type were required to be checked the phrase:

```
FOR-PLAYER  F-16
```

could be used which would check all players of the player-type F-16. All players which employ a LETHAL-ENGAGE-FIRING-START procedure may be examined by using:

```
FOR-PLAYER  ANYONE
```

Finally several players of different types may be selected by simply listing their names, for example we may have:

```
FOR-PLAYER  88 F-16
FOR-PLAYER  89 F-16
FOR-PLAYER  22 F-111
```

which would select the three players 88 F-16, 89 F-16 and 22 F-111 for debugging.

After specifying the procedure and the player or players whose tactics which will be checked it is necessary to specify which targets are relevant. (Since every RESOURCE-ALLOCATION procedure is designed to take action against some detected target, or targets, specifying here particular targets restricts the DEBUG command to occasions when targets of these types are involved.) In our example we have:

```
VS.-TGT  22 F-111
```

which selects the particular player 22 F-111 as a target. Just as with the selection of a player all players of a particular player-type:

```
VS.-TGT  F-111
```

or all players:

```
VS.-TGT  ANYONE
```

may be chosen. Finally a list of targets may be specified, just as in the case of selecting player names. For the RESOURCE-ALLOCATION procedures for which no particular target is relevant, i.e. emission control and launch start procedures, the target specifier is ignored and the keyword ANYONE should be used.

Once the above required entries have been set two optional filters may be used; the first, WITH-EMITTER, is effective with the four nonlethal engagement procedures: JAMMER-QUEUE-ADD, JAMMER-QUEUE-DROP, JAMMER-SPOT-ADD and JAMMER-SPOT-DROP. It narrows interest to cases where a target for jamming is a sensor transmitter or a communication transmitter of the indicated type, e.g.

```
DEBUG JAMMER-SPOT-ADD
  FOR-PLAYER  123 WILD_WEASEL_PLAYER
    VS.-TGT   121 SAM_BATTERY_PLAYER
    WITH-EMITTER  long_range_radar_tx
    USING-JMR     jukebox
END DEBUG
```

would debug attempts to jam the radar transmitter of type 'long_rang_radar_tx' using jammer's of the type 'jukebox'.

More generally the scope of the DEBUG command may be reduced to a portion of the scenario's duration with the TIME-WINDOW filter whose format is:

```
TIME-WINDOW  <start-time> <stop-time> <units>
```

which limits the debugging to the indicated period. In our original example:

```
TIME-WINDOW  15.0  55.0  (MIN)
```

the relevant period is from 15 to 55 minutes after the commencement of the scenario. The units used may also be seconds, (SEC), or hours, (HR).

The final entry in the DEBUG block must be present and identifies the nature of the resource that is either deployed or de-allocated by the player following the tactical procedure. Again there may be a list of resource names here if more than one possible outcome is to be considered. The form of this entry depends on the procedure being debugged; in our examples of debugging LETHAL-ENGAGE-FIRING-START and JAMMER-SPOT-ADD procedures the entries were respectively:

```
USING-WPN  ANYONE
```

and:

```
USING-JMR  jukebox
```

The relevant keywords for each of the RESOURCE-ALLOCATION procedures that can be debugged are summarized below:

Resource Allocation Procedural Class	`<using-spec>`	`<resource>`[26]
Lethal Assignment	USING-SUB	PLAYERS
Launch	USING-SUB	PLAYERS
Guns-Free/Guns-Tight	USING-SUB	PLAYERS
Non-lethal Engagement	USING-JMR	DISRUPTORS
Emission Control	USING-SNR	SENSOR RECEIVERS
Lethal Engagement	USING-WPN	WEAPONS

Once the DEBUG command has been fully specified and implemented it will produce output in the MOD listing file. This output lists those occasions where any criterion in a selected evaluation of the relevant RESOURCE-ALLOCATION procedures tested as false. It should be noted here that a single criterion may evaluate as false whilst the complete test evaluated as true; to see this consider the various possible formats of a RESOURCE-ALLOCATION logical test. The simplest possibility is that some single criterion is tested, for example:

```
2D-DIST < 43.0   (KM)
```

In this case if the criterion is false then this whole filter of the RESOURCE-ALLOCATION procedure will also be false and no resource will be allocated or de-allocated from any logical pipe using this filter. (Other filters in the same procedure leading to the use of some resource may still be true however.) Output of the form shown below will be issued by the DEBUG instruction:

```
88 F-16 FAILS LETHAL-ENGAGE-FIRING-START
  TARGET:  22 F-111
    WEAPON:  1040 AIM_LCHR
      CRITERION FAILED:  2D-DIST; FILTER 1
        CURRENT VALUE/THRESH:  47390.832  43000.000
```

This refers to the RESOURCE-ALLOCATION procedure LETHAL-ENGAGE-FIRING-START in the tactics block of the player 88 F-16. This player is engaging 22 F-111 using its weapon '1040 AIM_LCHR', but it cannot fire it due to the failure of one criterion. The criterion in question is that ensuring that the target is at most 43 km away found in the procedure's first filter, which fails because the current separation is 47.391 km.

[26] Resource identification is made from the appropriate list of names in the UAN. The input takes the form of the appropriate resource name optionally preceded by its SDB identification number or the phrase ANYONE.

A more complex test may combine several criteria, as in:

```
2D-REL-TGT-OFFSET < 49.5 (KM) OR 2D-DIST < 43.0 (KM)
```

This test may still be true even in the case of the second criterion being false, however the above output which lists the failure of the second criterion will still be given even though the whole statement may be true. If both criteria were false DEBUG output relating to both criteria will be given.

If a compound test is made with the AND statement, as with:

```
2D-DIST < 43.0 (KM) AND REL-TGT-HDG < 99.0 (DEG)
```

then the statement will be only true if both criteria are true. In this case information relating to the *rightmost false criterion only* will be given. So if the second criterion is true then the above output would be given by DEBUG, otherwise a statement relating to the REL-TGT-HDG criterion would be given instead.

In more complex cases such as:

```
2D-REL-TGT-OFFSET < 49.5 (KM) OR
2D-DIST < 43.0 (KM) AND REL-TGT-HDG < 99.0 (DEG)
```

the rules on DEBUG output are a combination of the above. Recall that AND always binds more closely than OR so this statement is equivalent to A OR (B AND C). The whole test will be true if either A or both B and C are true. Since, in the example we are considering, we know that B is false then what is printed out by the DEBUG statement depends on the truth of C. If C is true then the status of B will be printed, otherwise the status of C will be printed out instead. If A is false its status will also be given in both cases.

6.5 Monitoring Players During the Model Run

The data item MONITOR PLAYER allows one specified player to be monitored using the terminal's display screen during model execution. DISPLAY must be one of the specified options for OUTPUT:, and MONITOR PLAYER can only be used if the computer uses standard ASCII escape sequences. (The latter is always true for UNIX machines.)

Three entries are required to adequately describe the player that is to be monitored. Firstly, the player's identification number and name must be specified and these must correspond with a the number and name of a player defined in the `PLAYER:` data item of the SDB:

```
MONITOR PLAYER  <player-id> <player-name>
```

This is followed by the identification number of the player's location, again this must correspond with the number already specified in the `PLAYER:` data item of the SDB:

```
LOC:  <id-number>
```

The final entry is a time interval which defines the rate at which the output screen for `MONITOR PLAYER` will be updated:

```
EVERY <time> SECONDS
```

The time interval is entered as a positive real number measured in seconds, and refers to game time.

The output contains information on sensor acquisition and tracking, weapon firing, both what the player is doing and what it is undergoing from other players. Both an instantaneous and a cumulative count of the total number of certain events is shown on the screen. The instruction:

```
MONITOR PLAYER  22 F-111  LOC: 1  EVERY 5.0 SECONDS
```

in the example scenario produces the example output shown below at some point in the simulation:

```
                MONITORING    22 F-111   loc: 1
Fuel        19808.656 kg                     Now    Cum
X             100.246 km     Seen by:         0      3
Y              73.894 km     Tracked by:      0      1
Z (MSL)         2.483 km     Fired at:        0      2
Speed         277.800 m/s    Sees:
Heading       347.471 deg    Tracking:
Attitude        0.000 deg    Firing
```

As can be seen other information displayed about the monitored player includes its remaining fuel, its position in Cartesian coordinates, its speed and heading and attitude angle. Furthermore it reveals that it is not currently being seen by or seeing any other players, but that previously it has been seen and tracked and even shot at by other players.

6.6 Operational Availability

Apart from the selection of a random number seed the only way that the MOD data set can influence the outcome of a Suppressor simulation is with the `OPERATIONAL AVAILABILITY` item. This may be used to randomly remove players which are found to be unavailable, i.e. non-operational, at the commencement of the model. This feature is useful for modelling such things as mechanical failure or unanticipated malfunction of equipment. The command is somewhat limited, though, in that only players which are neither commanders nor movers may be made unavailable with this command. So it may not be used to eliminate any players which are moving or may move, nor any players which play a command role.

In the following example the respective probabilities that the two SAM will be functional are 0.65 and 0.99. In either case if a unit random variable selected by Suppressor exceeds the threshold the player will be removed from the scenario.

```
OPERATIONAL AVAILABILITY
  PLAYER TYPE  sam-a  AVAILABILITY  0.65
  PLAYER TYPE  sam-b  AVAILABILITY  0.99
END AVAILABILITY
```

The format can be seen from this example, a `PLAYER TYPE` line must exist for each player. The entry after the `AVAILABILITY` term gives the probability, (which clearly should be between zero and unity), that each player of this type will in fact be available during the scenario. Note that once a player is unavailable it will remain unavailable for the whole scenario.

6.7 Summary of the MOD stage

Once the MOD stage is complete the Suppressor scenario has in fact been run. The output data is then available as a record of the scenario and can subsequently be processed and analysed. The Suppressor simulation system provides one further processing stage, the Analysis Data Base or ADB, which can aid in the interpretation of the results. To make use of this the MOD binary file must be produced and retained for use by the ADB and an ADB instruction set must be created by the analyst. The ADB instruction set is discussed in the next section of the guide.

7. Analysis Data Base

The Analysis Data Base (ADB) is a Suppressor post-processing tool for use by the analyst. It allows the user to select situations of interest by filtering the output collected in the 'Time History List' during the model's execution. This filtering is achieved by defining criteria in much the same way as in the `RESOURCE-ALLOCATION` of the TDB. It should be noted that the ADB is not, unlike the other processing stages, an essential step in studying a scenario with Suppressor. It is quite possible to analyse the results of the model execution phase with other means, perhaps written expressly for that purpose by the analyst, and not use the ADB at all. In many cases however, the ADB will provide a convenient method of analysing the large amounts of data produced by Suppressor. It will often be especially useful in the early development of a Suppressor study, before more refined techniques, for example, the use of visualization methods, are employed. Furthermore the ADB should be useful in making an initial selection of the data for subsequent, and more careful, study.

The reminder of this section provides detailed information on the formatting requirements of the ADB instruction set as well as an introduction in the use of this comprehensive tool.

7.1 Default Output Formats

The ADB, like all other files, starts with the standard commands `EXECUTE` and `INSTRUCTIONS FOR:` which identify the data-set being processed as the ADB:

```
EXECUTE
INSTRUCTIONS FOR:
     ADB
```

These are then followed by four optional initialization variables and an embedded data item, `SITUATION:`, which can occur several times.

The first of the initialization variables allows the analyst to focus on a particular period of time during the model's run. There are two options: viz. `TIME-ALL` (which is the default) ensures data is collected throughout the entire scenario, and:

```
TIME WINDOW  <start>  <stop>
```

which defines the beginning and the end of a time frame. Time variables are entered as non-negative real numbers in units of seconds, note that unusually for Suppressor these units are not actually specified.

As part of the ADB processing step a count of the frequency of occurrence of each type of event will automatically be produced. These data can be collated into a histogram, with individual data bins of the histogram having widths specified with the `FREQUENCY-INTERVAL` command. This interval is defined in seconds, specifying a value of zero or the omission of the command will suppress production of the histogram. The selection of a non-zero value, for example:

```
FREQUENCY-INTERVAL  10.0
```

would yield histogram cells with widths of the specified number of seconds. In this case the histogram should be interpreted as showing the number of times the events occurred within each ten second interval which collectively span either the `TIME WINDOW` or the whole of the scenario.

The ADB processor will also produce a listing of all the time history data items that meet the criteria set within the `SITUATION:` item. This listing can be suppressed with the entry `LISTING OFF`.

The final global variable is the page width:

```
PAGE WIDTH  <number>
```

which defines the character width of the printed page. The model default is 80 characters and again this can be specified in either the ADB or the embedded data item.

7.2 Formatting of the `SITUATION:` Data Item

Individual situations of interest to the analyst can be identified with the `SITUATION:` data item, of which at least one must be given. Generally several of these will be included in the ADB input file. Each one consists of a comment line, followed by any of the four optional variables described above as global options, and a precise description of the situation. They are entered into the file using the following format:

```
SITUATION:  <comment>
  <options>
  {situation-description}
END SITUATION
```

The comment is a required entry allowing the situation to be named and a short description to be given. If the comment follows the `SITUATION:` keyword on the same line it cannot extend onto further lines. Should a longer comment spanning several lines be desired it must commence instead on the line following the `SITUATION:` keyword. Examples of this are:

```
SITUATION:  All_lethal_assignments_made_by_side_Blue
```

and for a longer comment:

```
SITUATION:
All_cases_of_friendly_fire_for_side_Blue_during_the
later_part_of_the_scenario_when_Blue_fighters_are
providing_air_support_for_the_2nd_bomber_wing
```

No spaces may be included as part of the comment and, as with other Suppressor input files, no line may extend beyond 72 characters in length.

Any of the global variables described above, `TIME-ALL`, `TIME WINDOW`, `FREQUENCY-INTERVAL`, `LISTING OFF` and `PAGE WIDTH`, can be overridden within each `SITUATION:` entry. The only complication to watch for is if a global `TIME WINDOW` has been specified for the ADB along with a `FREQUENCY-INTERVAL` in order to construct a histogram of events, for example:

```
TIME WINDOW            3600.0  5400.0
FREQUENCY-INTERVAL       60.0
```

In the above case if then a local `TIME WINDOW` is specified in a `SITUATION:` entry which is not completely contained within the global time window, for example:

```
TIME WINDOW            3000.0  4000.0
```

then a local `FREQUENCY-INTERVAL` must also be defined, for example:

```
FREQUENCY-INTERVAL       50.0
```

Each specified situation is made up of one or more sentences which themselves describe specific events. These sentences are linked together with the phrase OR-ALSO since the situation will be true if any of the sentences composing it are true. The sentences are constructed of components which follow rules based on English grammar. Each ADB sentence consists of one or more nouns, at least one predicate (which is also known as a verb-phrase) and may also include one or more prepositions, which are one of the words ABOUT, BY, FOR or FROM. The nouns are either a subject, i.e. the name of the player taking the action (the nominative case), an object, the name of the player against whom the action is taken (the accusative case), or an indirect object, i.e. the name of a player who is involved indirectly in the event (the dative case). The nouns will be one of the following:

- ANYONE i.e any player in the scenario,
- ALL <side> PLAYERS i.e. any player on the scenario side whose name 'side' is listed in the SDB.
- ANY <player-type> is any of player of the player-type with name 'player-type' identified in the TDB.
- THE <id-no> <player> is the uniquely identified player with identification number 'id-no' and name 'player' found in the SDB, e.g. '23 F111'.

These nouns can be qualified with one or more location selection phrases:

- AT LOCATION <loc-id> to specify that the noun refers to the players' elements found at the relevant location, as specified in both the SDB and TDB.

Multiple AT LOCATION phrases, of which any can be used to select the relevant elements, can be used by connecting them together with the OR keyword.

Sentences used in the SITUATION: entry always begin with the subject and are then followed by at least one predicate. The predicates are known also as verb-phrases and describe the action being undertaken by the subject. For example a very simple sentence is:

```
ANYONE  LEAVES-A-WAKEUP-CALL
```

which would select any instance of any player leaving a wakeup call for a move plan. (This action, along with most of the others available in the ADB, is described in the reference manual.) The above sentence requires no object since the action, 'leaving a wakeup call', is not directed against any particular entity. More than one subject can appear connected by the keyword OR:

```
THE 32 F-111 AT-LOCATION 10 OR
   ALL Orange PLAYERS FIRST-INDIRECTLY-SEES  ANYONE
```

The above example selects instances when either the specified mover or any player on the 'Orange' side indirectly sees any target, i.e. learns of the target through intelligence received rather than through its own sensors. The targets, represented by the noun ANYONE, are this sentence's direct object. Just as with the subject of the sentence more than one object can be linked with the use of the keyword OR. An example of this is:

```
ANY sam_player SUCCESSFULLY-HIT THE 32 F-111 OR 33 F-111
```

Here the action could have been taken against either of the two specified players. Multiple predicates can also be linked with the use of the OR connector, yielding examples such as:

```
ANY bomber-a STARTS-MOVEMENT OR STOPS-MOVEMENT
```

Some verb-phrases require the use of indirect objects, for example:

```
ANYONE INTENDS-TO-INFORM ALL Red PLAYERS ABOUT ANYONE
```

Here the subject is ANYONE, the predicate is INTENDS-TO-INFORM, the object is ALL Red PLAYERS and the indirect object is ANYONE. The indirect object is always preceded by a preposition, which, as noted above, is always one of the keywords ABOUT, BY, FOR or FROM.

The tables below list all the verb phrases along with whether or not they take a direct object or indirect object, and, where appropriate, the preposition used. All predicates need a subject. Most of the prepositions correspond directly to items found in the Time History List, the only exception being the special predicate DOES-ANYTHING-TO which can be used to select any action taken by a subject against an object. The required presence of a direct object is shown with a '•' symbol, where an indirect object is also required this is shown by the specification of the appropriate preposition.

The first table lists just those predicates which take neither an object nor a preposition.

Predicate	Predicate
,ALT-CMDR,-SEIZES-CONTROL	SCRAMBLES-TO-ORBIT
CHANGES-TERRAIN-FOLLOWING-ALTITUDE	SELF-DESTRUCTS-IN-FAILURE
COASTING-BEYOND-PATH-END	STARTING-TO-CHANGE-NET-FREQ
CRASHES-INTO-THE-GROUND	STARTING-TO-CHANGE-RADAR-FREQ
CUES-HEADING-TOWARD-TARGET	STARTING-TO-RELOAD
DEPARTS-ORBIT-FOR-A-PATTERN	STARTS-MOVEMENT
DID-NOT-CHANGE-ITS-PATH	STARTS-TO-EXECUTE-A-PLAN
DISCONTINUES-MANEUVER	STARTS-TO-HEAD-TOWARDS-CHECKPOINT
EMPLOYS-A-VERTICAL-PROFILE	STARTS-TO-USE-PREDICTED-INTERCEPT-MODE
FINISHED-CHANGING-RADAR-FREQ	STARTS-TO-USE-PURSUIT-INTERCEPT-MODE
FINISHED-RELOADING	STOPS-MOVEMENT
FOCUSES-A-PREEMPTIVE-JAMMER-SPOT	STOPS-TERRAIN-FOLLOWING
GETS-MODE-OF-CONTROL-CHANGE	SUSPENDS-MOVEMENT
HAS-LETHAL-ENGAGE-AUTHORITY	TURNS-OFF-COMM-TRANSMITTER
HAS-NO-OPERATIONAL-SUBS	TURNS-OFF-IMPLICIT-JAMMER
HEADS-HOME,-OUT-OF-ACTION	TURNS-OFF-JAMMER-TRANSMITTER
IGNORES-OUTDATED-PLAN	TURNS-OFF-SENSOR-RECEIVER
INTENDS-TO-USE-LAST-ROUND	TURNS-OFF-SENSOR-TRANSMITTER
IS-RETURNING-TO-SDB-PLANNED-PATH	TURNS-ON-COMM-TRANSMITTER
LEAVES-A-WAKEUP-CALL	TURNS-ON-IMPLICIT-JAMMER
LEAVES-AN-ORBIT-FOR-A-POINT	TURNS-ON-JAMMER-TRANSMITTER
LOST-LAST-SUB-TO-ATTACK	TURNS-ON-SENSOR-RECEIVER
MANEUVERS-TO-PLAN-ASPECT-ANGLE	TURNS-ON-SENSOR-TRANSMITTER
POSSIBLY-MIGHT-CRASH-LATER	UPDATES-INTERACTIONS
REACHES-CHECKPOINT	USING-TRACKER-FOR-ACQ-ALSO
RESETS-CUING-TO-DEFAULT	WAKES-UP-TO-THINK-ABOUT-PLAN
RESUMES-MOVEMENT	WILL-NOT-STOP;-IN-ORBIT
RUNS-OUT-OF-GAS	WILL-START-PATTERN-MOVEMENT
	WILL-THINK-OF-PRESET-PLAN

The second table lists those commands which take an object but no preposition.

Predicate	Predicate
ABANDONS-SALVOING-AGAINST	IGNORES-OLD-DETECTION-OF
ABORTED-INFLIGHT-SHOT-AT	,IN-ENVELOPE,-INTERCEPTS
ADDS-ENTRY-TO-JAMMER-QUEUE-FOR-TGT	INITIATES-LOCKON-OF
ADDS-SUB-TO-ASGN-QUEUE-FOR-TGT	INITIATES-PERCEPTION-OF
ADDS-WPN-TO-ENG-QUEUE-FOR-TGT	IS-ASSIGNED
ADJUSTS-JAMMER-SPOT-FOCUSED-AT	IS-JAMMED-(EXPLICITLY)-WHILE-SENSING
BAFFLED-BY-ASG-CANCEL-FOR	IS-TRANSMITTING-MESSAGE-TO
,BY-COMMAND,-ENGAGES	LOCKON-SIGNAL-LOSS-FROM
CAN'T-MANEUVER-AGAINST	MANEUVERS-IN-RESPONSE-TO
CAN'T-USE-NEW-DETECTION-OF	MANEUVERS-TO-ATTACK
CAN-CONTINUE-SALVO-AGAINST	MOURNS-DEATH-OF
CANNOT-INTERCEPT	,NET-BUSY,-WANTS-TO-TALK-TO
CHANGE-IN-DETECTION-FOR	,NET-FREE,-WANTS-TO-TALK-TO
CHANGE-IN-SENSING-STATUS-FOR	NO-LONGER-PERCEIVES
,COASTING,-STOPS-FIRING-AT	,NOT-IN-ENVELOPE,-INTERCEPTS
DECENTRALIZES-CONTROL-TO	NOW-SENSING-ELEMENTS-OF-TGT
DECIDES-TO-SHOOT-AT	,ON-ITS-OWN,-ENGAGES
DELETES-JAMMER-QUEUE-ENTRY-FROM-TGT	RANDOMLY-LOSES-LOCK-ON
DIGESTS-RESULTS-OF-ATTACK-ON	REMOVES-A-JAMMER-SPOT-FROM-TGT
DISCONTINUES-TRACKING-OF	SEES-DEATH-OF-KNOWN
DOES-ANYTHING-TO	SENDS-SCRAMBLE-TO
DROPS-SUB-FROM-ASGN-QUEUE-FOR-TGT	SHOT-SPENT,-UNAWARE-OF-DEATH-OF
DROPS-WPN-FROM-ENG-QUEUE-FOR-TGT	STOPS-ENGAGEMENT-OF
EXPECTS-TO-INTERCEPT	STOPS-SENSING-CHANCES-FOR
FIRES-A-WEAPON-AT	SUCCESSFULLY-HIT
FIRST-DIRECTLY-SEES	TERRIBLY-PUZZLED-BY-MSG-FROM
FIRST-HAS-SENSOR-IN-RANGE-OF	TOLD-OF-NO-AMMO-BY
FOCUSES-A-JAMMER-SPOT-ON-TGT	UNFORTUNATELY-MISSED
GETS-AN-ASSIGN-CANCEL-FOR	UPDATES-(W/SNR)-DATA-ON
GIVES-UP-TRYING-TO-TALK-TO	WILL-BE-OUT-OF-RANGE-OF
HAD-A-BAD-LAUNCH-AGAINST	WILL-TRY-AGAIN-TO-TALK-TO
HAS-LOST-CONTACT-WITH	WON'T-USE-OLD-LOCATION-DATA-FOR-TGT
HAS-NO-ZONE-AUTHORITY-FOR	YEARNS-TO-SEND-ASG-UPDATE-TO

Finally the next table indicates those verb phrases which include an object and a preposition.

Predicate	Preposition
CONFUSED-BY-ASG-STATUS-FROM	FOR
DELETES-ASSIGNMENT-TO	FOR
FIRST-INDIRECTLY-SEES	FROM
INTENDS-TO-INFORM	ABOUT
IS-BACKED-UP,-TRIES-TO-TELL	ABOUT
PICKS,-FOR-LETHAL-ASG,-SUB	FOR
SENDS-ASSIGN-CANCEL-TO	FOR
SENDS-DEATH-NOTICE-TO	ABOUT
TOLD-ABOUT-DEATH	BY
TOLD-OF-DEATH-OF-KNOWN	BY
UPDATES-(W/MSG)-DATA-ON	FROM
WANTS-TO-TELL-NEW-STATUS-TO	ABOUT

Complex events can be selected by combining together multiple nouns and predicates as building blocks within each `SITUATION:` data item. Often the same result can be achieved in more than one way, for example the sentence:

```
ANY F-18 OR ANY F-111
,IN-ENVELOPE-INTERCEPTS OR SUCCESSFULLY-HIT
ANY F-16 OR THE 60 airbase
```

is identical in effect to the two sentences linked with `OR-ALSO` below:

```
  ANY F-18 OR ANY F-111 ,IN-ENVELOPE-INTERCEPTS
  ANY F-16 OR THE 60 airbase
OR-ALSO
  ANY F-18 OR ANY F-111 SUCCESSFULLY-HIT
  ANY F-16 OR THE 60 airbase
```

In this case the former would be simpler to use. However, problems start to arise when using verb phrases that take different types of objects, in these cases two separate sentences linked by `OR-ALSO` must be used. An example of this would be:

```
  THE sam_player PICKS,-FOR-LETHAL-ASG,-SUB ANYONE
  FOR ANY F-18 OR ANY F-111
OR-ALSO
  THE sam_player FIRST-DIRECTLY-SEES ANY FA-18 OR ANY F-111
```

In the above example we are selecting events in which either the '`sam_player`' is assigning a subordinate against the named fighters, (here indirect objects), or first perceives them, (where the fighters are direct objects). Because the two sentences use different types of objects they cannot be linked into one sentence using a single structure. The same problem would arise if two predicates both took indirect objects but used different prepositions, or if one predicate took no object at all.

7.3 Output of the ADB

The ADB instruction set is the final processed stage of the Suppressor run. As noted in this guide's overview no binary file is produced by this stage, but rather merely the listing file whose contents are selected by the ADB instructions described above. Output corresponding to the situation:

```
SITUATION: Successful_hits_by_Blue_on_Blue
  LISTING OFF
  ALL Blue PLAYERS SUCCESSFULLY-HIT ALL Blue PLAYERS
END SITUATION
```

is given below:

```
==> Begin Situation: Successful_hits_by_Blue_on_Blue
*+*  MODEL MESSAGE 80 (INFORMATIVE): Summary of Incidents
Frequency of Occurrence for Situations:
SUCCESSFULLY-HIT                              4
==> End Situation: Successful_hits_by_Blue_on_Blue
```

Because the listing option was off the Time History list for these events is not given; if this were not so the items would be recorded in the same manner as they appeared in the MOD listing file.

With the completion of the ADB processing step the Suppressor simulation is complete. Other analysis tools, such as graphical post-processors and user supplied programs may of course be used in completing the study. The main role of these tools is to facilitate the extraction of Measures of Effectiveness (MOEs) from the results of the simulation. In practice several different such measures could be obtained from a single study of a complex scenario, each involving different ADB files or other analysis tools, but utilizing the same MOD output.

8. Conclusions

This guide should serve as both an introduction and reference to the Suppressor Simulation System which, as has been shown, is a comprehensive and flexible computer code for the modelling large scale military operations. Suppressor's great strength lies in its ability to model the interactions of systems whose properties can be defined by the user. Because of this flexibility the systems modelled with Suppressor can employ Australian equipment in an Australian operational context. Furthermore these studies encompass not only the characteristics of specific physical systems but also the command, communication, control and intelligence (C^3I) structures present in the scenario.

Suppressor is well suited for examining the overall picture of a mission level model, which allows the focus to be drawn away from asking how well particular systems perform in isolation to how well they perform in integrated environments. In other words how well can specific military hardware perform in local C^3I structures when used together with existing materiel. Note that in addition to testing the efficacy of specific equipment Suppressor can indeed be used to examine how changes in the C^3I structure can maximize the effectiveness of military systems. With these insights a clearer picture can be drawn of the advantages, or disadvantages, of using or acquiring complex military equipment.

Suppressor's ability to model responses to dynamically changing and so inherently unpredictable situations means that simulations of complex missions are feasible, allowing likely weak points and difficulties to be predicted. This aspect of large scale operational modelling, that is the simulation of the unpredictable elements which can and do go wrong, gives Suppressor some considerable usefulness. Of course, Suppressor cannot predict when things will break down, or unexpectedly come good, but rather can be used to see what happens should these events occur and in what circumstances these events are likely. Suppressor is perhaps not so well suited for modelling routine and easily predictable operations, e.g. a study of the effectiveness of routine patrols would not be a good use of Suppressor's capabilities. On the other hand, such modelling activity can be useful for calibrating the results of other, more complex scenarios.

The Suppressor Simulation System is held by the Air Operations Division in executable form only, which makes verification of the system by exercising different parts of the source code impracticable. However confidence in the results must be held for credence to be given to studies completed using Suppressor. Fortunately two avenues are available for checking the accuracy of models built using Suppressor. The first, and most straightforward, is the process of cross checking the results of a Suppressor simulation with those of another model, in whose results the analyst has confidence. Unfortunately this technique cannot be used to confirm the correctness of those parts of a Suppressor study which can be only be handled by Suppressor. Since presumably the reasons for using Suppressor to model a particular scenario were precisely to deal with these aspects of the problem this is a significant difficulty. A second, and better method, would be to validate the model using the type data base in a scenario with known outcomes, i.e. to use Suppressor to model the results of a military exercise employing as many of the scenario's component systems as possible. This approach, although certainly a non-trivial one, would provide a great deal of insight into how well Suppressor performs with these systems and the potential shortcomings, inadequacies and pitfalls of the model.

Suppressor was designed explicitly for modelling aerial warfare, although it can and does model ground and naval forces which are involved in such actions. Also, due to Suppressor's enormous flexibility, it is quite capable of handling situations and systems well beyond the scope of its initial design aims. From the Australian perspective the areas in which Suppressor models may well be appropriate include the following:

- Airborne Early Warning and Control: Suppressor's ability to model sensor, communication and command systems should make it a useful tool for studying the effectiveness of AEW aircraft and systems.
- Air Defence of Australia: Suppressor is capable of handling large scenarios and so is quite suitable for modelling the integrated capabilities of ground, air and sea based warning and interceptor systems, along with the efficacy of surface to air missiles.
- Maritime Patrol: Maritime Patrol is a closely related field to AEW, albeit with different aims and the significant extension that the patrolling aircraft usually cover a much wider area and have a strike capability. Again, Suppressor is very suitable for handling the major components of such scenarios although it cannot handle sub-surface detections, i.e. no sonar physics is included in the model.
- Maritime and Strategic Strike: In addition to defensive scenarios Suppressor can model strike actions, including strike missions against both maritime, ground and air targets. In an integrated model strike and defensive missions are just two different sides of the same coin, since each model can be seen from both viewpoints.
- Stand-Off Missiles: With its ability to spawn `FUTURE-PLAYER`s as fully fledged model entities Suppressor can provide detailed models of guided missiles. These include both passively and actively guided missiles in addition to models of anti-radiation ARM missiles designed to destroy hostile radar and surface-to-air missile sites.
- System Procurements: Suppressor can model the benefit accruing when major physical systems are upgraded; these would include any sensor, weapon, electronic counter measure equipment or airframe upgrades or replacements.
- Command, Communication, Control and Intelligence: As well as being a tool for modelling physical systems in a given C^3I environment Suppressor can equally well be used to model the robustness of C^3I structures when involved in some physical environment, or to study a combination of the two factors. This allows studies to draw conclusions about not only the type of systems employed but also to consider how they are used.

Because of Suppressor's ability to re-use information pertaining to the capabilities and tactics of its TDB's component systems in many different scenarios attention needs to be given to the problem of compiling relevant data-bases for the Suppressor TDB. These data should describe both ADF systems and those of potential adversaries. This information, concerning as it does many different systems and platforms, will form a significant corpus of information for use in modelling of large scale operations of the types discussed above.

As an indication of both the kind and the amount of data required for a scenario suppose that a model of a strike mission against an airfield is to be constructed. (From another point of view this is also a model of the defence of an airfield.) Obviously a TDB containing information about all the relevant aircraft, of perhaps three or four types, needs to be constructed. These might include bombers of the attacking side, fighters of types flown by both combatants along with perhaps command and control and defence suppression aircraft. In addition to the physical capabilities of these aircraft information relating to their sensing, communication, weapons and ECM equipment borne by all these sytems as well as stationed on the ground needs to be modelled at a fine level. The airfield's ground defences, such as various kinds of missile batteries and radar sites, must also be included. All the systems in the model, including perhaps stand-off weapons such as actively guided missiles, will need tactics defined for their operation which are flexible enough to encompass all conceivable eventualities. In addition the capabilities of the various decision making systems, i.e. the human players and computers involved in the scenario, must be defined in terms of how many there are of them, what their responsibilities are as well as how long they take to carry them out. Furthermore the chains of command and control, along with the various sorts of messages that are exchanged along them, the details of communication frequencies used and the times required to pass these messages must all be defined. As well as the above material detailed topographic information must be gathered to allow modelling of terrain masking and the flight paths of aircraft which fly low to avoid detection. The movement and contingency plans and the initial briefing information of all the components of the model must be provided in full. It can be seen that a major operational model is a far from trivial undertaking, especially if TDB instruction sets have to be compiled from scratch.

Suppressor is a medium level simulation system, designed for the modelling of individual operations at a mission level. This means that it differs from larger, campaign models such as THUNDER, in that it does not allow the incorporation of several missions spanning many days into a single model. However Suppressor can be used as a vital adjunct in the modelling of very large scale conflicts lasting many days or weeks. Suppose that a analyst wishes to model in some detail the prosecution of a one week conflict involving two or three major operations on each day of the campaign. Each individual mission can be modelled as a Suppressor scenario with the results obtained from these models being used as input for the campaign model, e.g. THUNDER. Campaign models have the ability to span many days and include the effects of system degradation and the limitations imposed by logistical considerations. The combined effect of such a model would be to give detailed insights into the implications on a campaign resulting from tactical decisions and command structures employed at the mission level. A future aim of integrated mission and campaign level modelling should be to examine not only models such as Suppressor and THUNDER in isolation, but also to examine the difficulties and advantages inherent in linking them together in such 'meta-models'.

Suppressor is a valuable tool for modelling medium and large scale military operations in an Australian context. It allows the modelling of many different complex scenarios using an essentially constant set of characteristics which define the component systems taking part in these scenarios. Because a high order of physical fidelity can be captured by the model highly discriminatory measures of effectiveness (MOEs) can be deduced from these scenarios. These MOEs can in turn be used to analyse the effectiveness of component systems themselves, or the way in which they are used and controlled, or some combination of the above. Finally the results of Suppressor's mission level models can be combined to provide detailed models of entire military campaigns lasting from days to weeks when they are used with other, campaign level, models. In this way Suppressor can be used to play a part in an overall model of 'Short Warning Conflicts', an area of considerable importance to Australian defence.

9. Index of Key Words

DISTRIBUTION LIST

A Guide to using Suppressor in Studies of Large Scale Air Operations

Robert Whitehurst, Victor Kowalenko and Jane Phipps

AUSTRALIA

DEFENCE ORGANISATION

S&T Program

Chief Defence Scientist ⎫
FAS Science Policy ⎬ shared copy
AS Science Corporate Management ⎭
Director General Science Policy Development
Counsellor Defence Science, London (Doc Data Sheet)
Counsellor Defence Science, Washington (Doc Data Sheet)
Scientific Adviser to MRDC Thailand (Doc Data Sheet)
Director General Scientific Advisers and Trials/Scientific Adviser Policy and Command (shared copy)
Navy Scientific Adviser
Scientific Adviser - Army
Air Force Scientific Adviser
Director Trials

Aeronautical and Maritime Research Laboratory
Director

Electronics and Surveillance Research Laboratory
Director

Chief of Air Operations Division
Neil Pollock, AOD
Robert Dancer, AOD
Simon Goss, AOD
Graeme Murray, AOD
Gary Kemister, AOD
Serena Steuart, AOD
Helen Pongracic, AOD
Robert Whitehurst, AOD
Victor Kowalenko, School of Physics, University of Melbourne
Jane Phipps, AOD
Ken Anderson, TOG, Melbourne
Russell Pope, EWD
Nick Hadjinicolaou, EWD
Leigh Powis, MRD
Shane Brunker, EWD

DSTO Library
Library Fishermens Bend
Library Maribyrnong
Library DSTOS (2 copies)
Australian Archives
Library, MOD, Pyrmont (Doc Data sheet only)

Forces Executive
Director General Force Development (Sea)
Director General Force Development (Land)

Army
ABCA Office, G-1-34, Russell Offices, Canberra (4 copies)

S&I Program
Defence Intelligence Organisation
Library, Defence Signals Directorate (Doc Data Sheet only)

B&M Program (libraries)
OIC TRS, Defence Central Library
Officer in Charge, Document Exchange Centre (DEC), 1 copy
*US Defence Technical Information Centre, 2 copies
*UK Defence Research Information Center, 2 copies
*Canada Defence Scientific Information Service, 1 copy
*NZ Defence Information Centre, 1 copy
National Library of Australia, 1 copy

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy
Library
Head of Aerospace and Mechanical Engineering
Senior Librarian, Hargrave Library, Monash University
Librarian, Flinders University

OTHER ORGANISATIONS

NASA (Canberra)
AGPS

ABSTRACTING AND INFORMATION ORGANISATIONS
INSPEC: Acquisitions Section Institution of Electrical Engineers
Library, Chemical Abstracts Reference Service
Engineering Societies Library, US
Materials Information, Cambridge Science Abstracts
Documents Librarian, The Center for Research Libraries, US

INFORMATION EXCHANGE AGREEMENT PARTNERS
Acquisitions Unit, Science Reference and Information Service, UK
Library - Exchange Desk, National Institute of Standards and Technology, US

OUTSIDE AUSTRALIA

Brent Gindelberger,
Survivability Vulnerability Information Analysis Center
Booz-Allen & Hamilton Inc.
WL/FIVS/SURVIAC, Bldg. 45
2130 Eighth St., Suite 1
Wright-Patterson AFB, OH 45433-7542, USA

Bill McBride,
Effectiveness Analysis Section,
Naval Air Warfare Center Weapons Division
1 Administration Circle
China Lake, CA 93555-6001, USA

SPARES (10 copies)

Total number of copies: 71

Page classification: UNCLASSIFIED

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE A Guide to Using Suppressor in Studies of Large Scale Air Operations	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION) Document (U) Title (U) Abstract (U)
4. AUTHOR(S) Robert Whitehurst, Jane Phipps and Victor Kowalenko	5. CORPORATE AUTHOR Aeronautical and Maritime Research Laboratory PO Box 4331 Melbourne Vic 3001

6a. DSTO NUMBER DSTO-GD-0129	6b. AR NUMBER AR-010-153	6c. TYPE OF REPORT General Document	7. DOCUMENT DATE February 1997

8. FILE NUMBER M1/9/152	9. TASK NUMBER DST 94/211	10. TASK SPONSOR DSTO	11. NO. OF PAGES 242	12. NO. OF REFERENCES -

13. DOWNGRADING/DELIMITING INSTRUCTIONS None	14. RELEASE AUTHORITY Chief, Air Operations Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS

Mission (computer program), simulators, computer codes, modelling

19. ABSTRACT

This guide is an introduction and reference to the Suppressor Simulation System, a powerful and flexible computer code which allows models of integrated military operations to be put into an Australian context. Such mission level models are vital ingredients to models of entire military campaigns. This guide describes in detail how such models are constructed using Suppressor and also provides a comprehensive reference to the Suppressor command language to assist its use in AOD.

Page classification: UNCLASSIFIED

GENERAL DOCUMENT DSTO-GD-0129 AR-010-153 FEBRUARY 1997

DSTO
AUSTRALIA

AERONAUTICAL AND MARITIME RESEARCH LABORATORY
GPO BOX 4331 MELBOURNE VICTORIA 3001
AUSTRALIA, TELEPHONE (03) 9626 7000